

Code Versioning

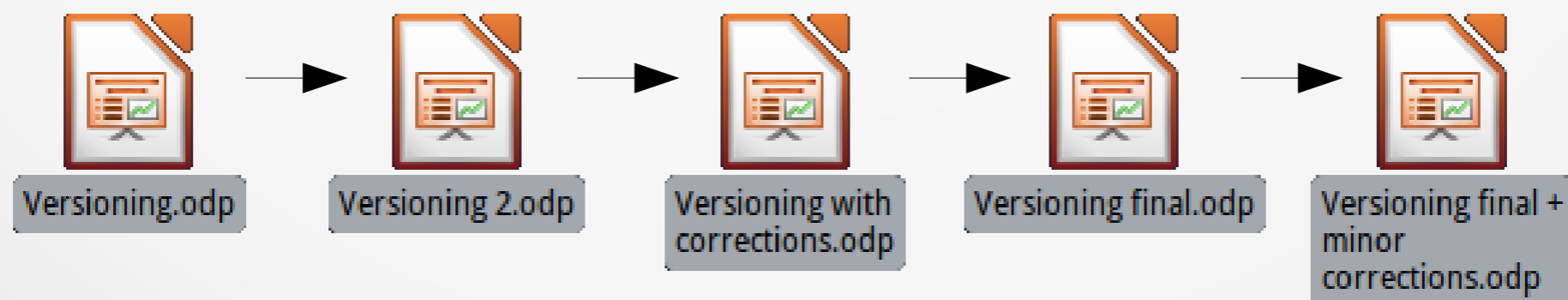
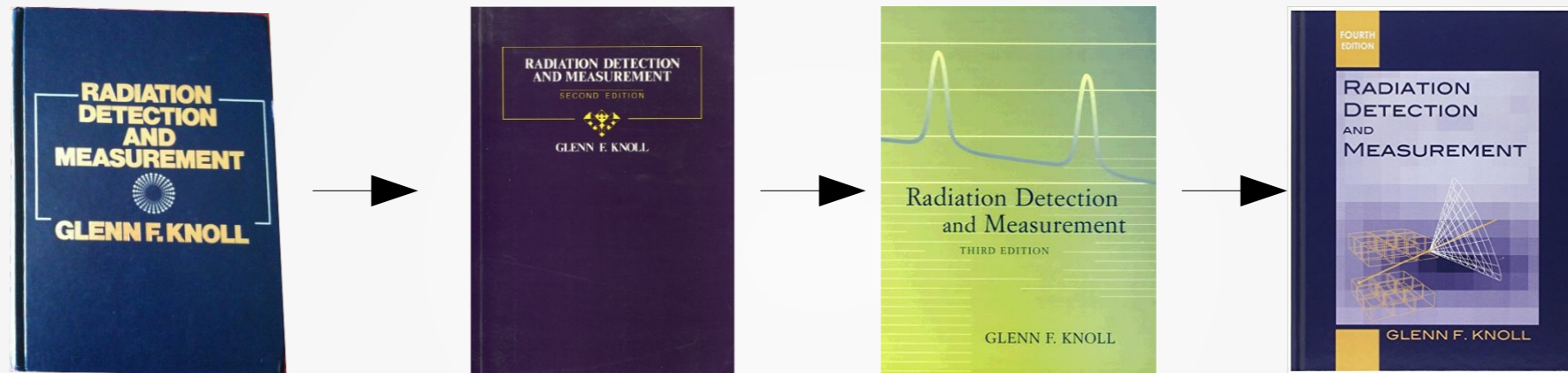
Olivier Mattelaer (CISM)

based on slides from
Damien Francois (CISM)
Juan Cabrera (NAMUR)
Jonathan Lambrechts (IMMC)
Scott Chalcon (git)

Road Map

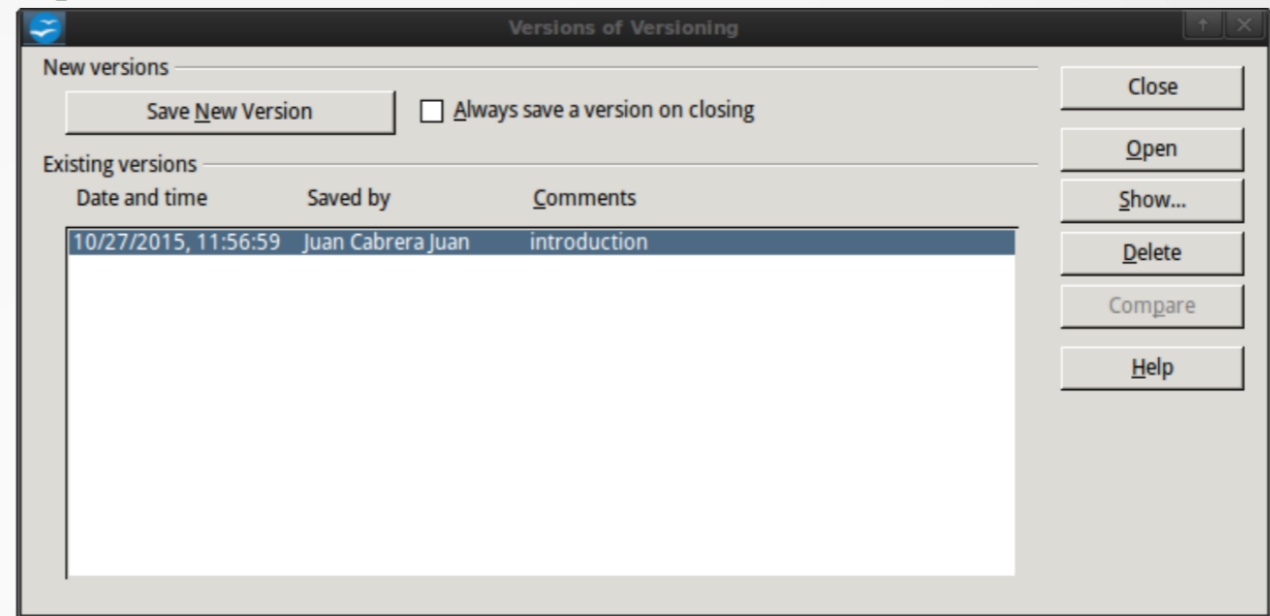
- historical perspective
 - various method of code versioning
- Basic of code versioning
 - revision, branch, conflict
- Single user: practical case
- Collaborating
 - Workflow
 - gitlab and similar

What is code versioning



Every day code versioning

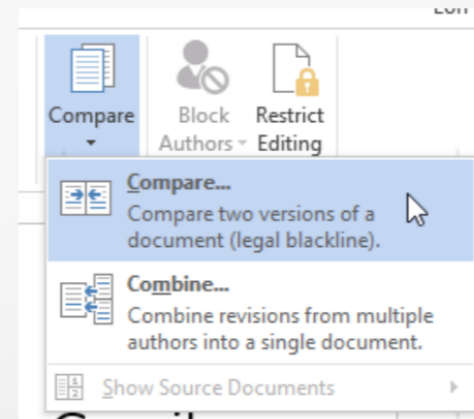
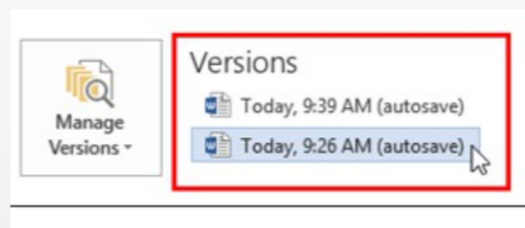
OpenOffice Documents



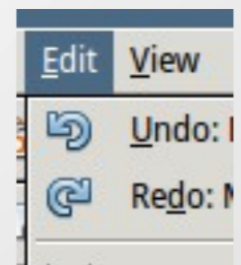
Google Docs



Microsoft Office documents



Undo-Redo



Goal of code versioning

1. History of modification
2. Team Work
3. WorkFlow

Goal of code versioning

1. History of modification

- Possibility to go back in time
 - Undo mistake / debugging /...
- Information about the modification
 - Who
 - When
 - Why

Goal of code versioning

2. Team Work

- **Simultaneous** work on a project
 - No need to send email to say “I’m working on that file” (dropbox organization)
- **Asynchronous** synchronisation
 - Allow work Offline (opposite to overleaf project)
 - Need conflict resolution

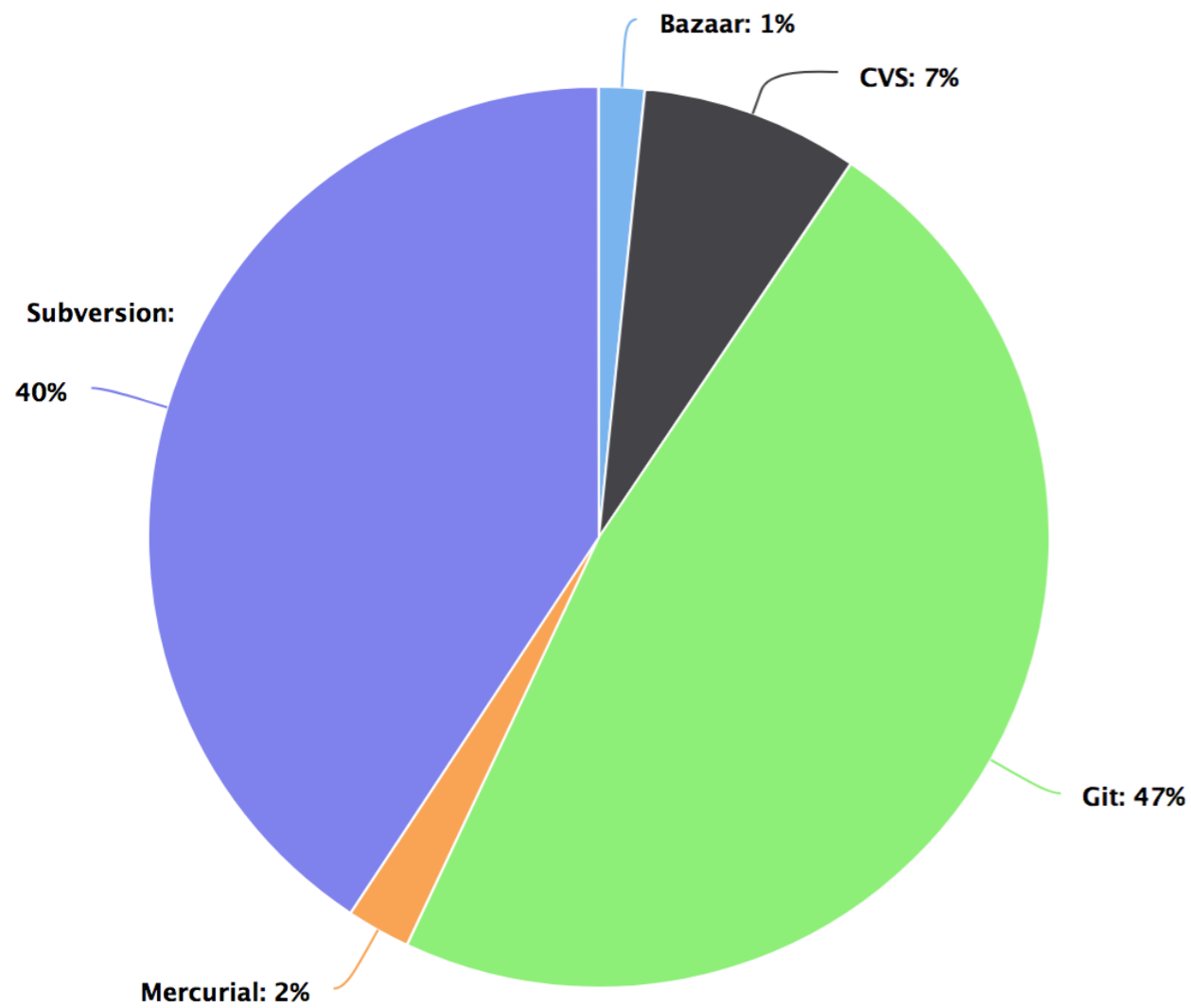
Goal of code versioning

2. Workflow

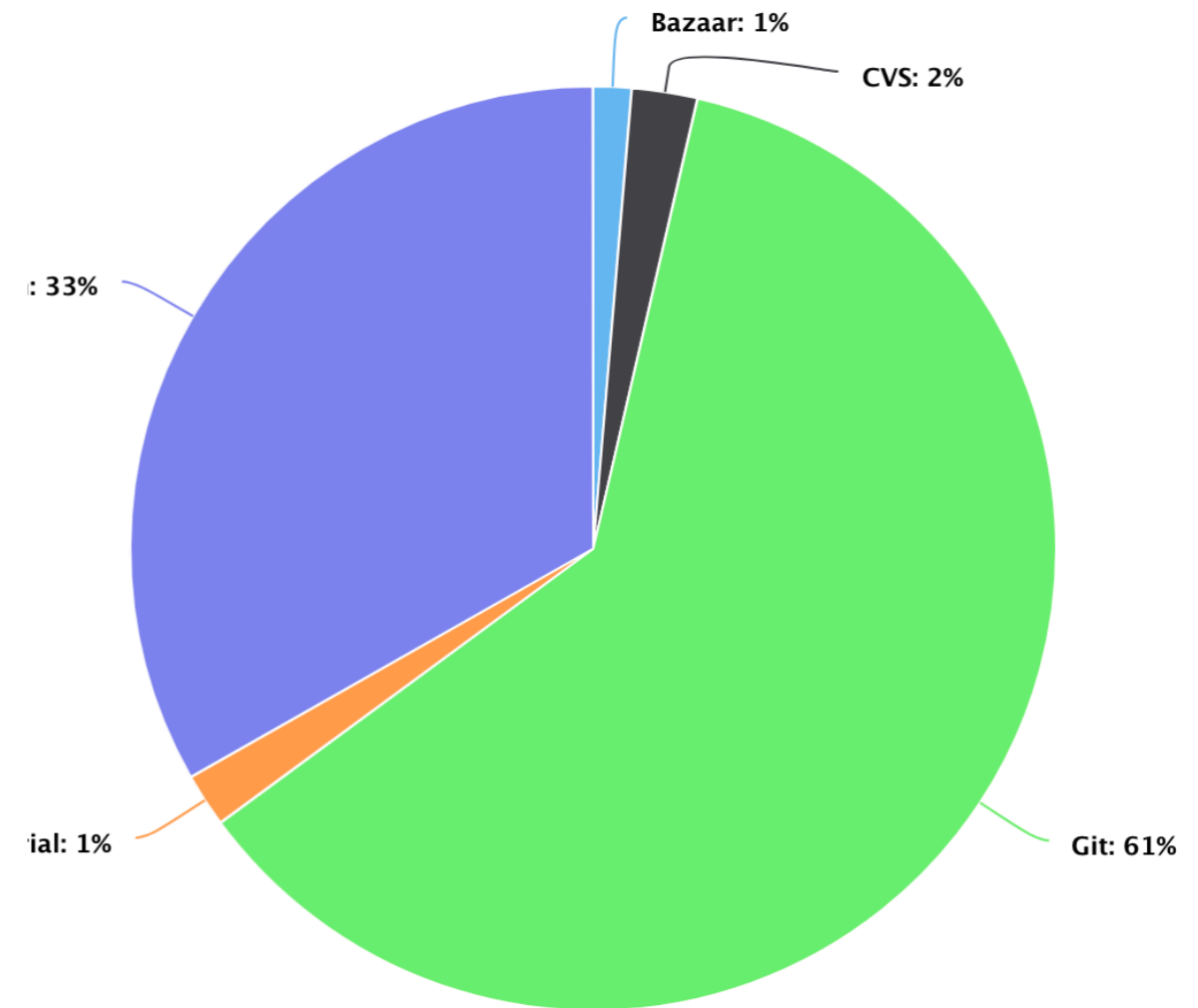
- **Testing new idea** (and easy way to throw them out)
- **Multiple version** of the code
 - Stable (1.x.y)
 - Debug (1.x.y+1)
 - Next “feature” release (1.x+1.0)
 - Next “huge” release (2.0.0)
- Need to pass modification from one version to next
 - Transfer of information between version

Open-Source Code

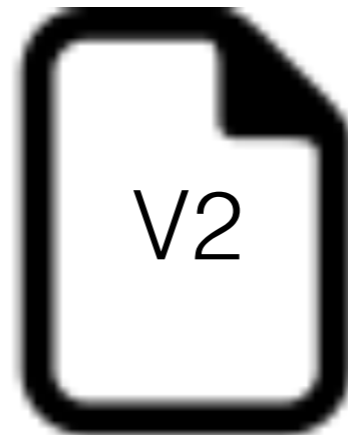
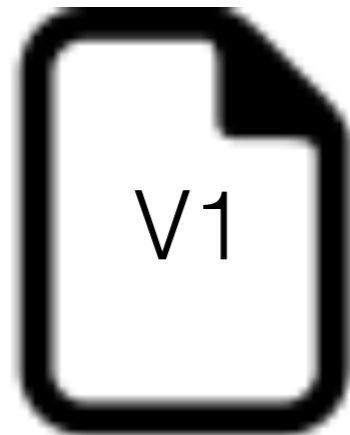
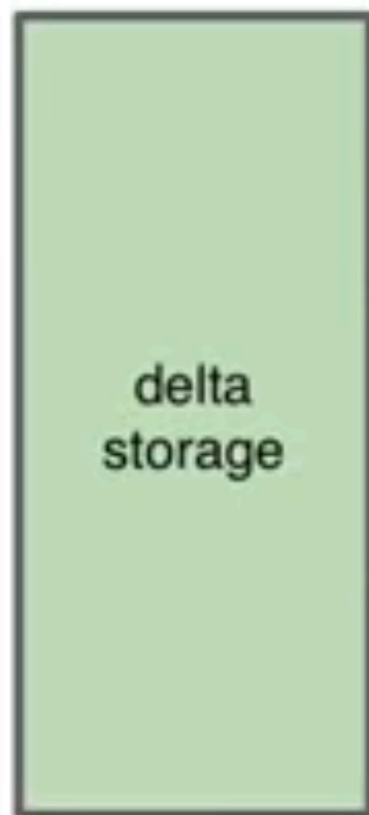
2017



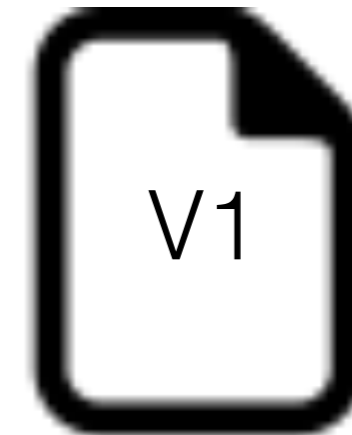
2018



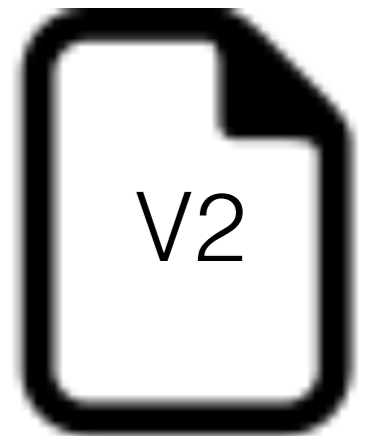
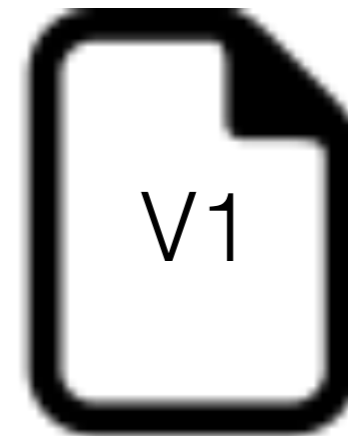
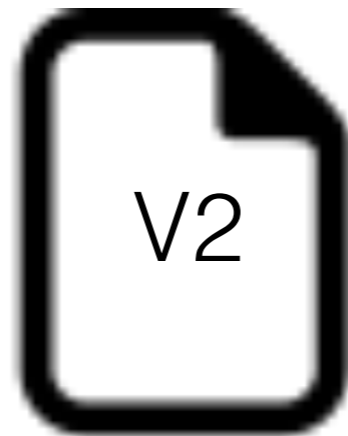
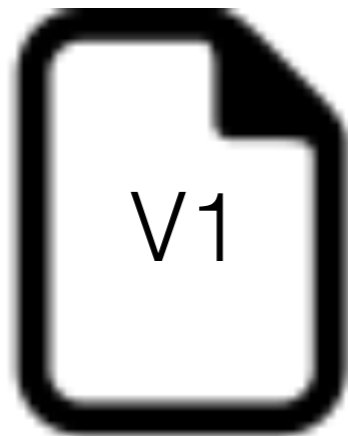
source control taxonomy



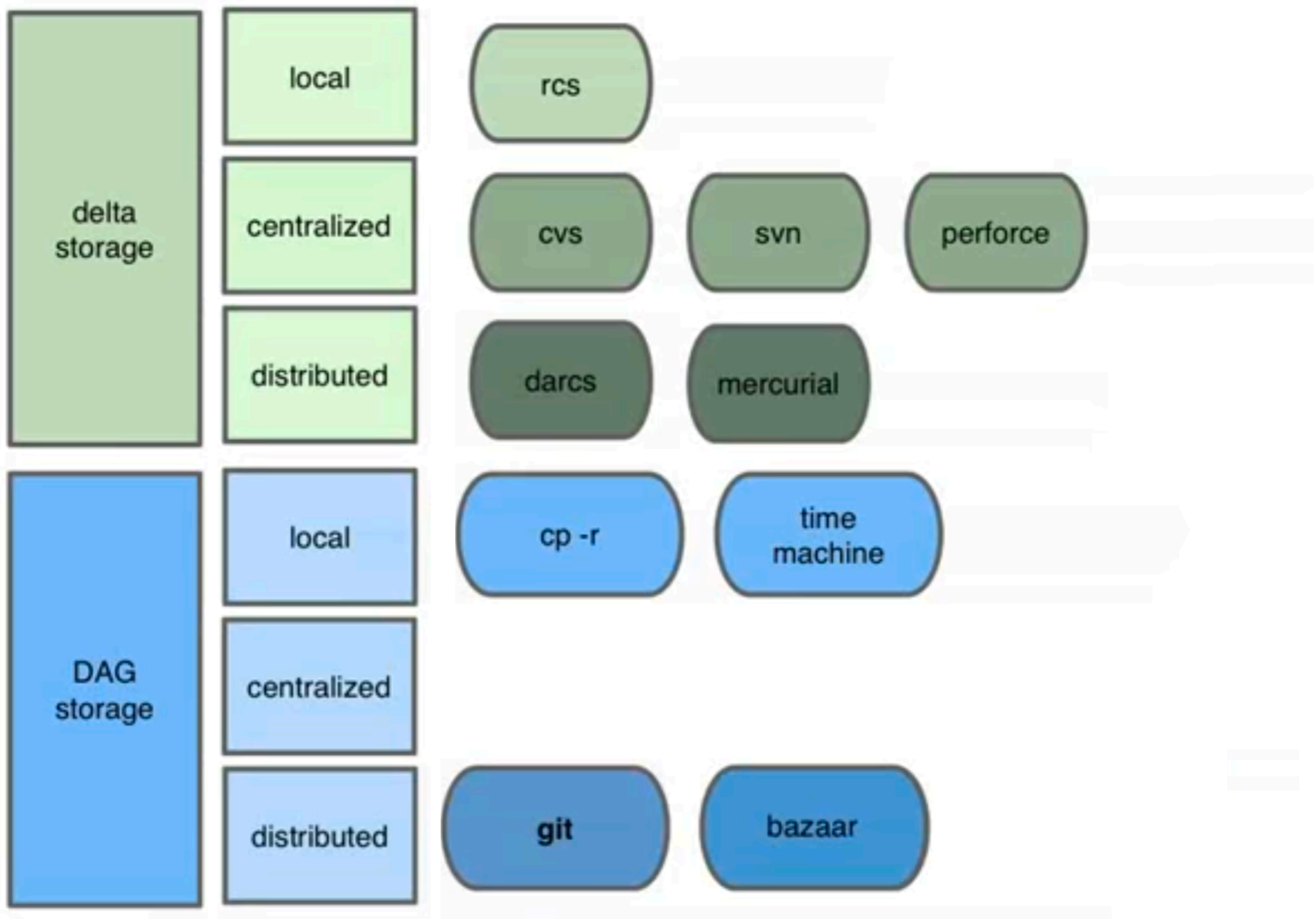
Repository content
Internal storage



Δ_{12}



source control taxonomy

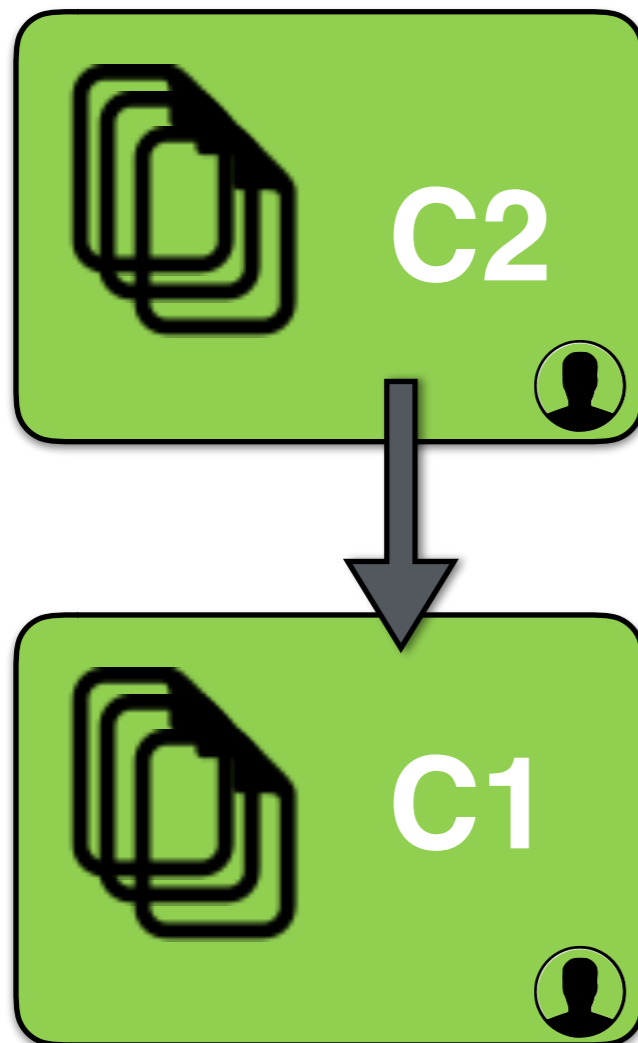


Key Concept

1. History
 1. History and commit
2. Three phases of git
 1. Workspace
 2. Index
 3. Repository

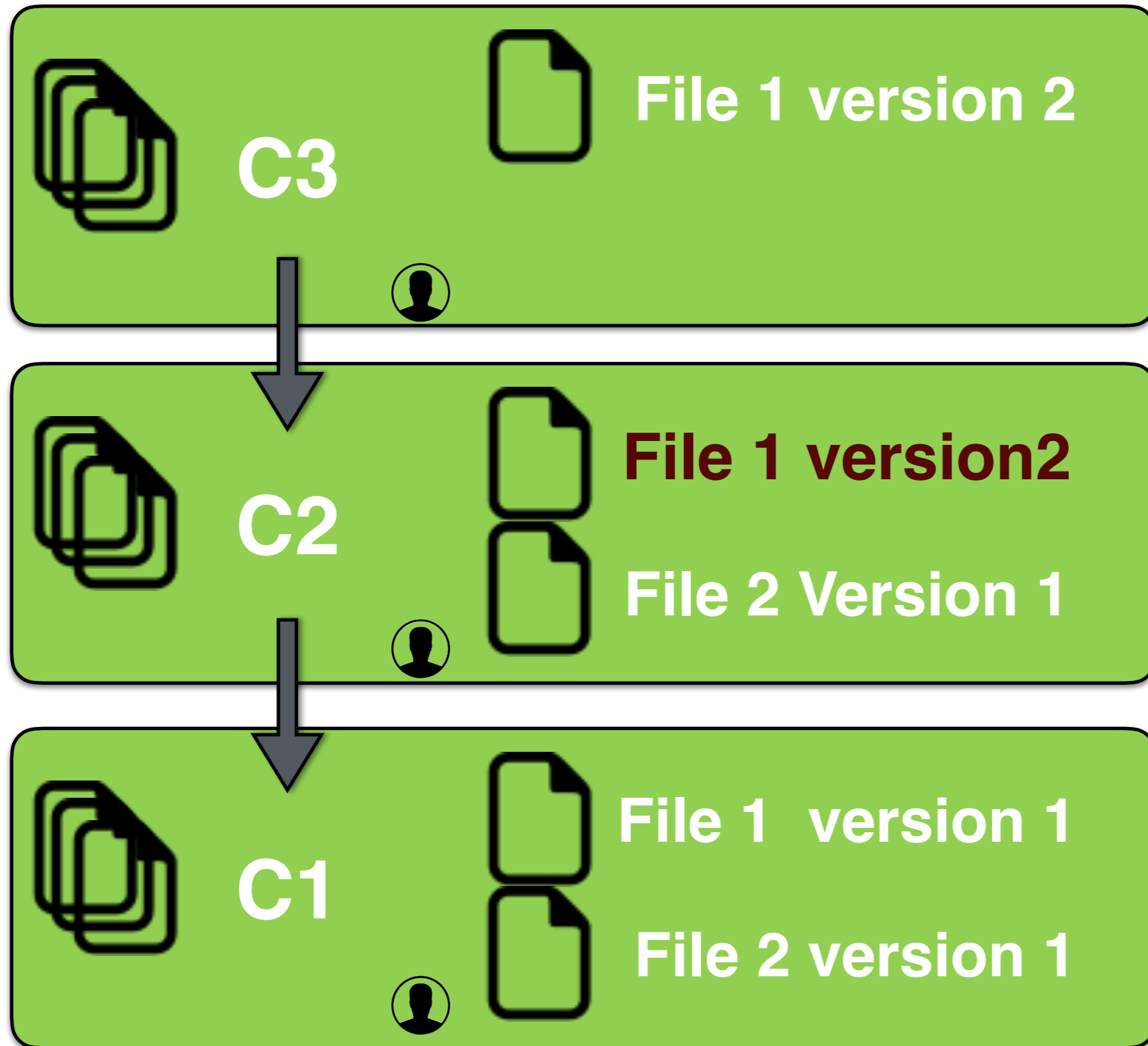
1. Commit

- An history: Is a **sucesion** of **snapshot** of your files at key time of their development
 - Each **snapshot** is called **COMMIT**



- Commit is
 - All the files at a given time
 - A unique name (SHA1)
 - MetaData (who created/when/info)
 - Pointer to previous(es) commit

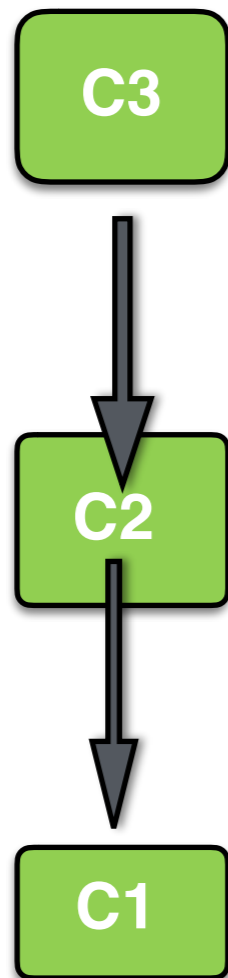
1. Commit



Remove file 2

Edit file 1

1. Commit



1. Simplify representation of commit/history

Git Three area

Workspace



./WORKDIR

Index



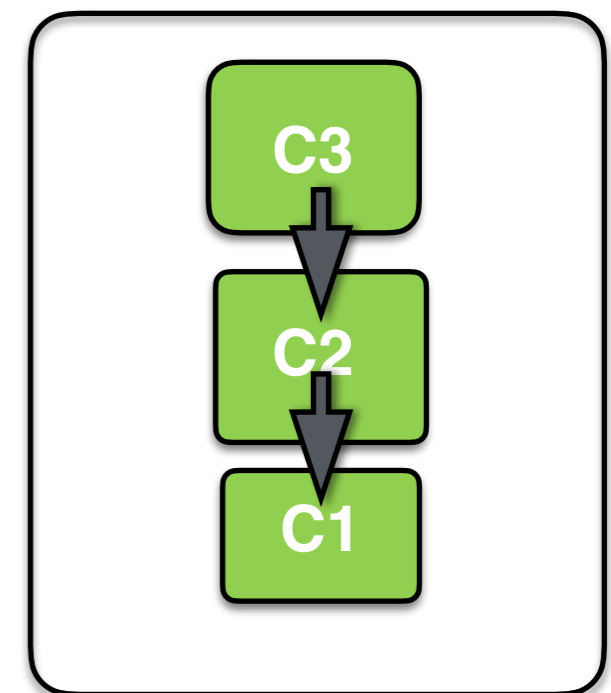
.git/index

Staging area

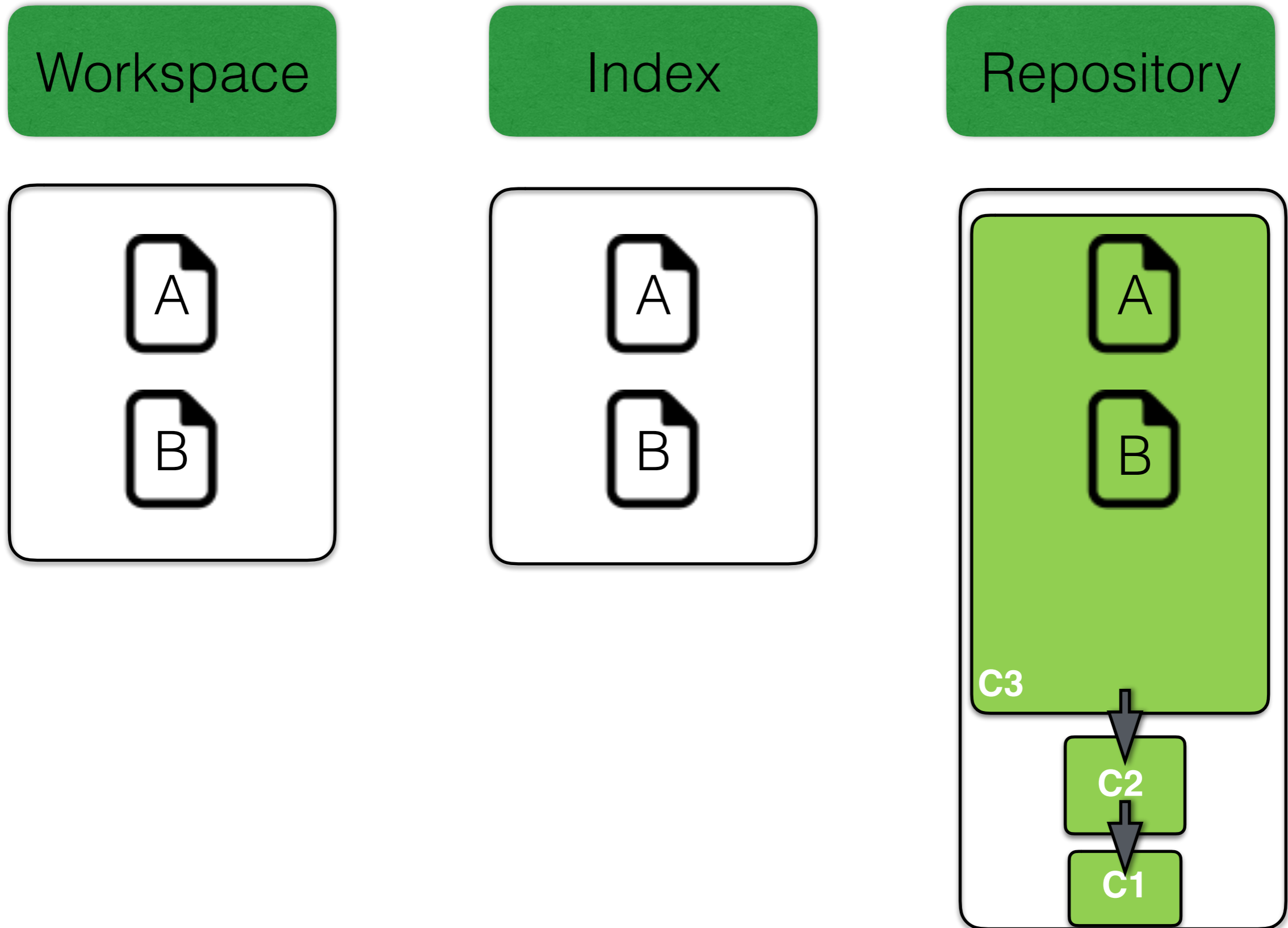
Repository



.git/



Git Three area

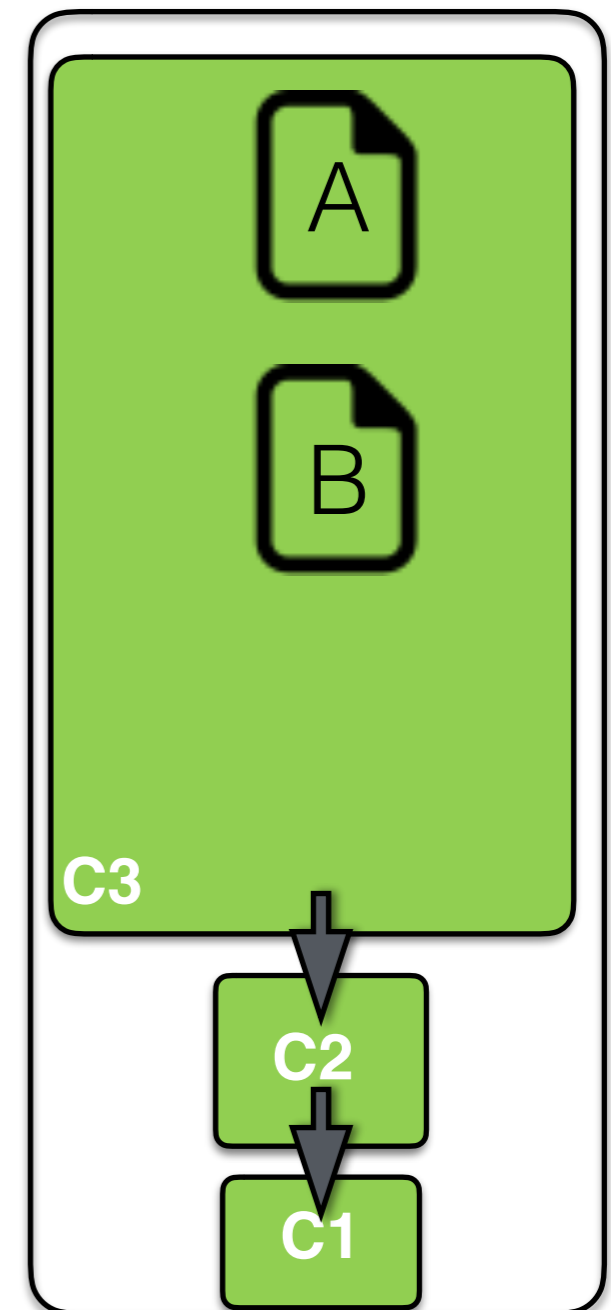
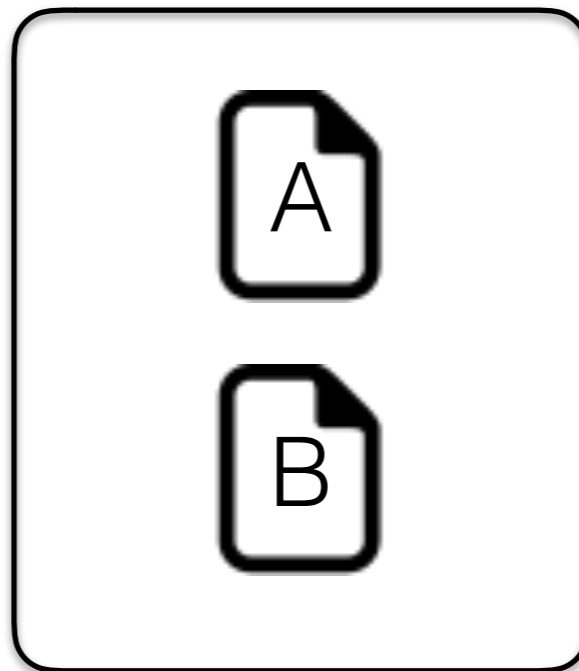
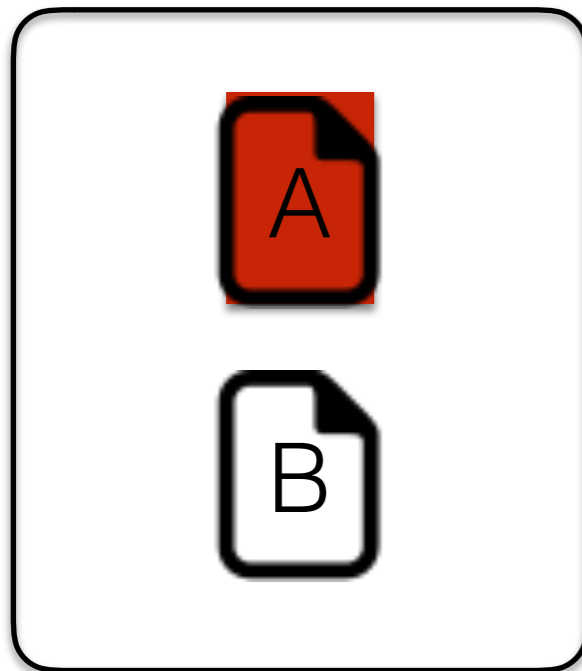


Git Three area

Workspace

Index

Repository

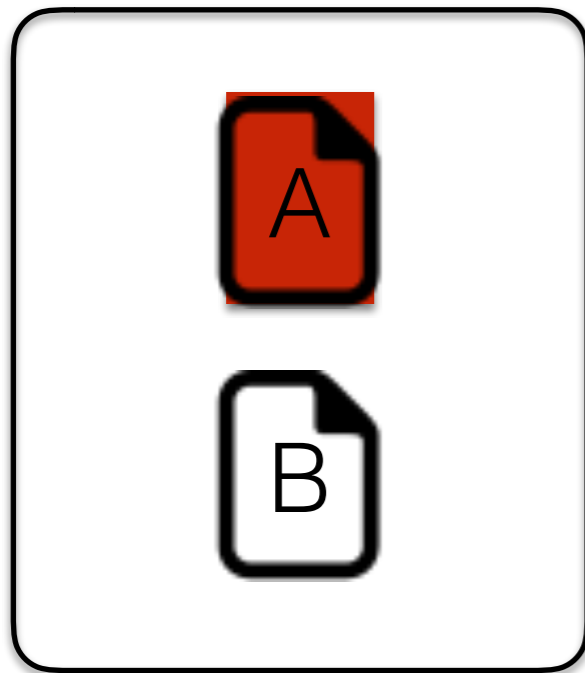


Action:

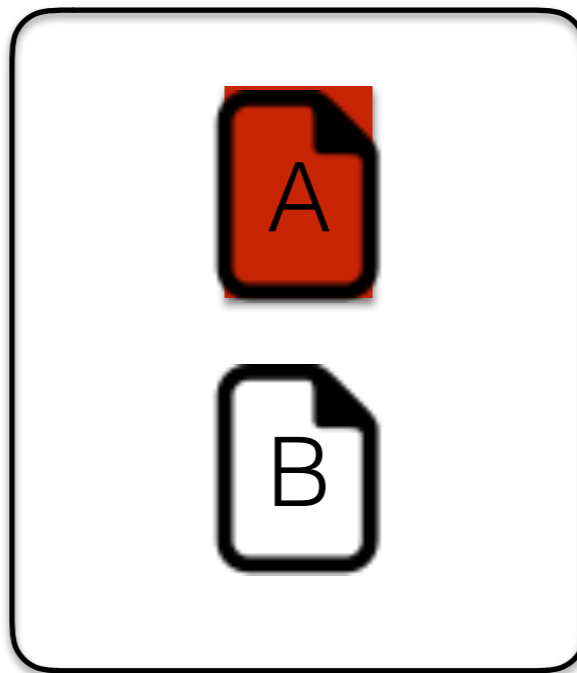
**Modifying file A
-> add a line**

Git Three area

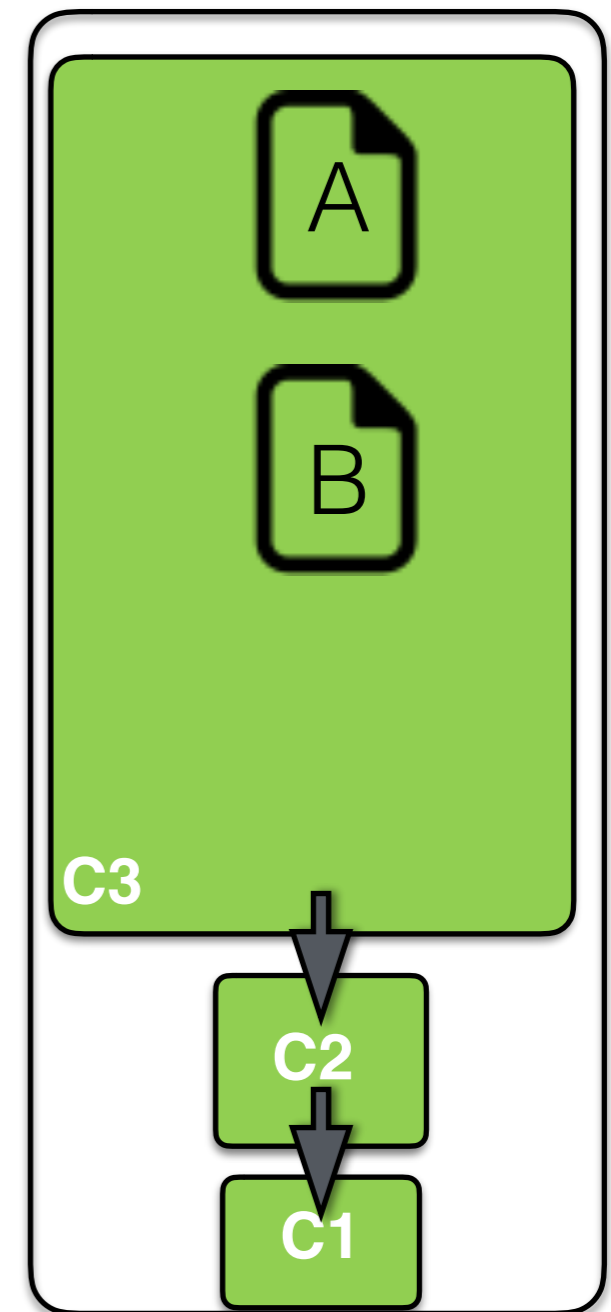
Workspace



Index



Repository



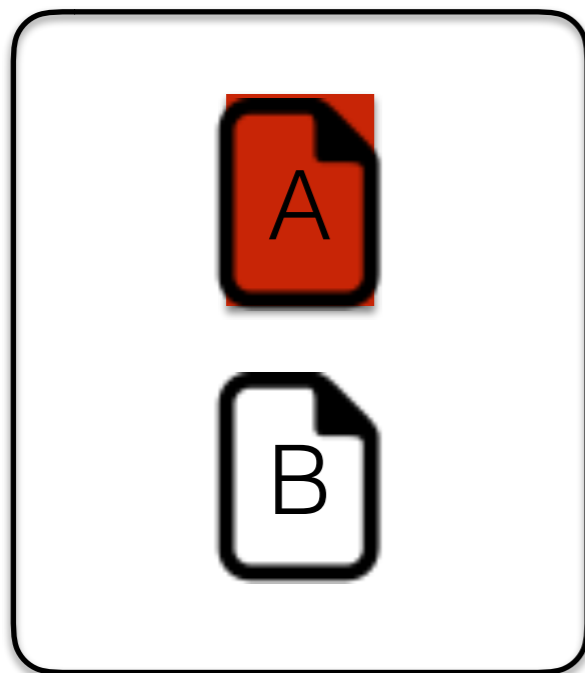
Action:

git add A

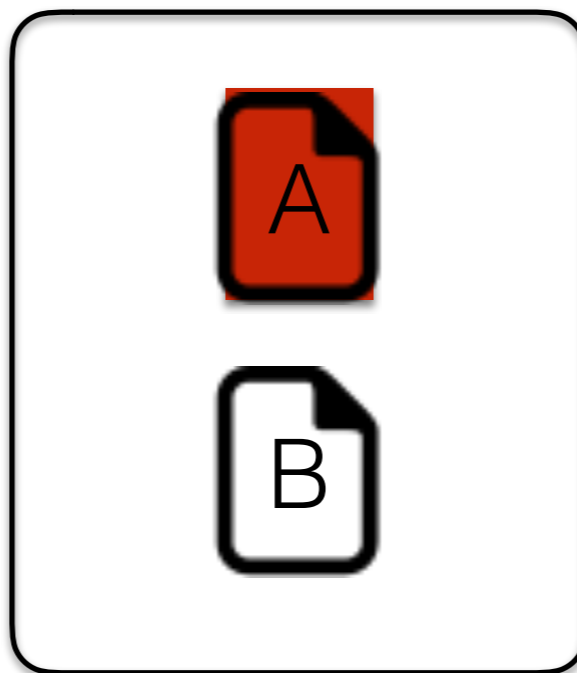
- > modify file moves to the index
- > inside the box
- > ready for a commit

Git Three area

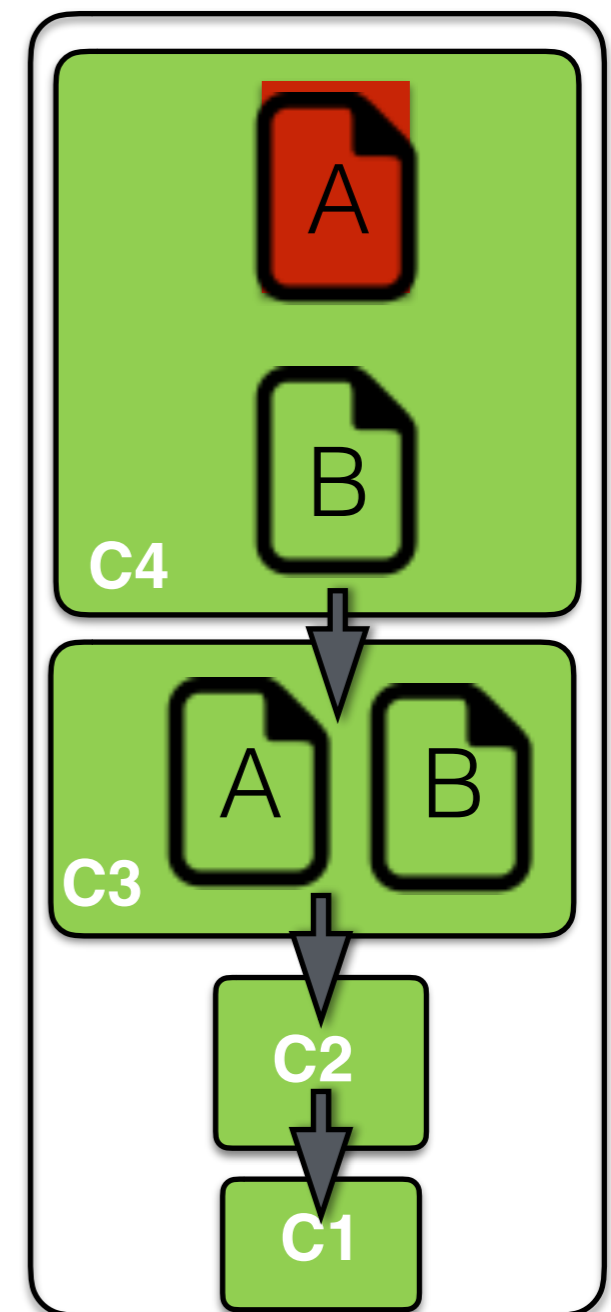
Workspace



Index



Repository

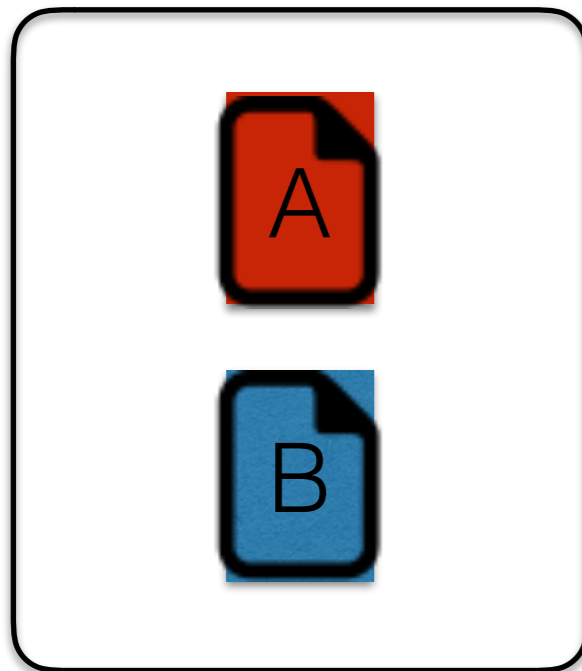


Action:

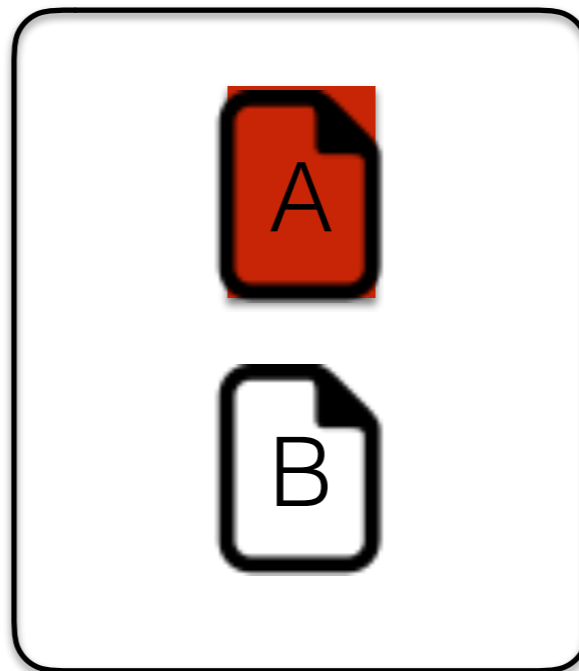
Git commit -m "change color"
-> save the index current status
Into a new commit inside the
Repository

Git Three area

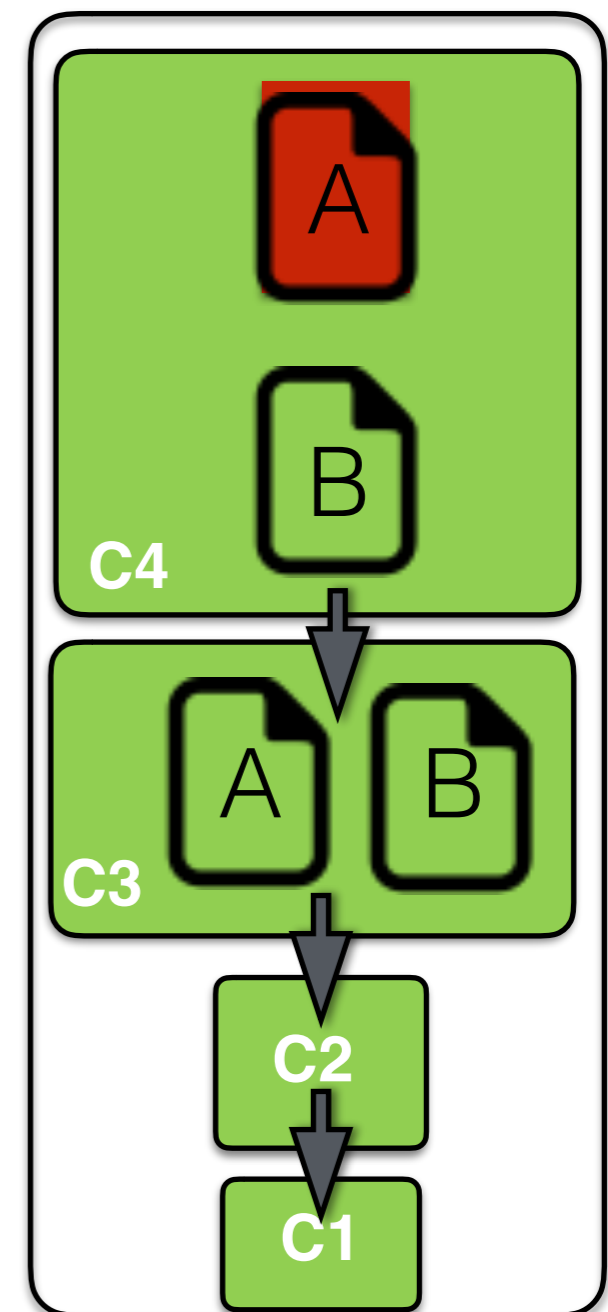
Workspace



Index



Repository



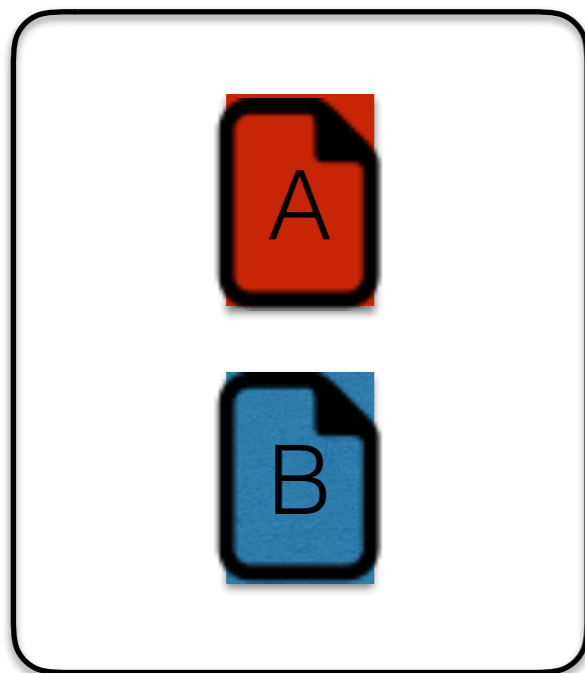
Action:

Edit file B

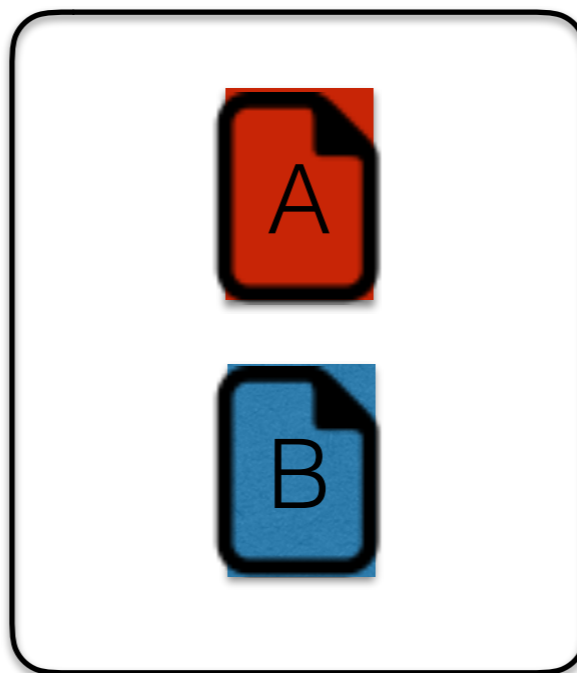
git commit -a -m "second one"

Git Three area

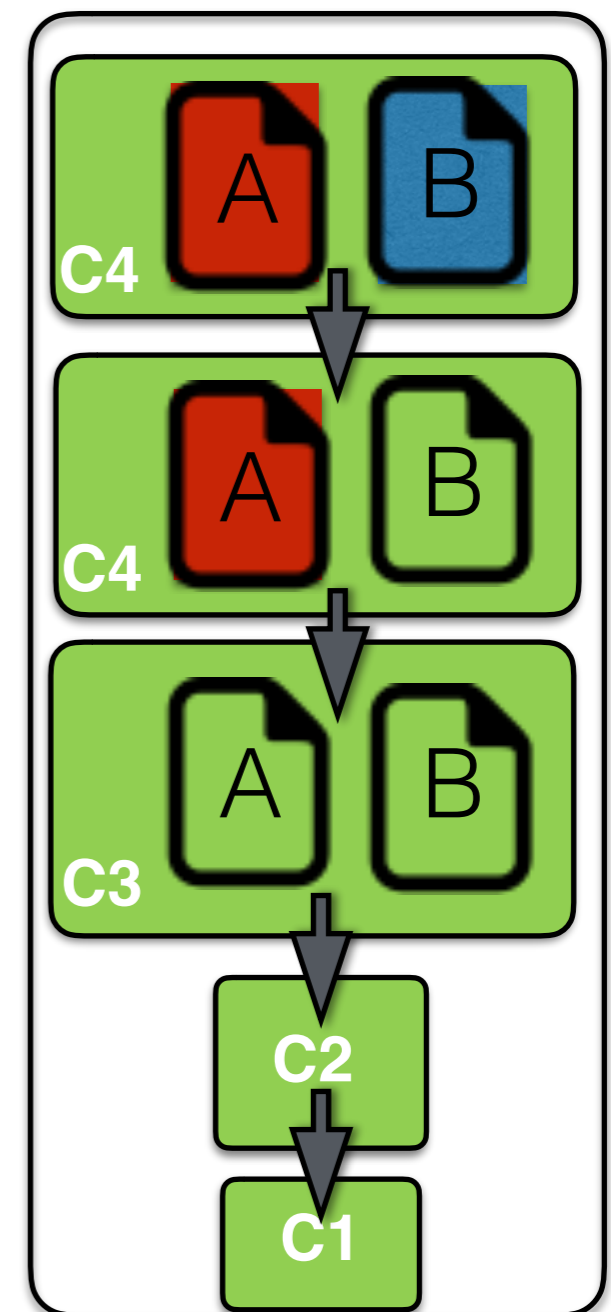
Workspace



Index



Repository



Action:

git commit -am "change color2"
-> automatic staging of edited
file and removed file

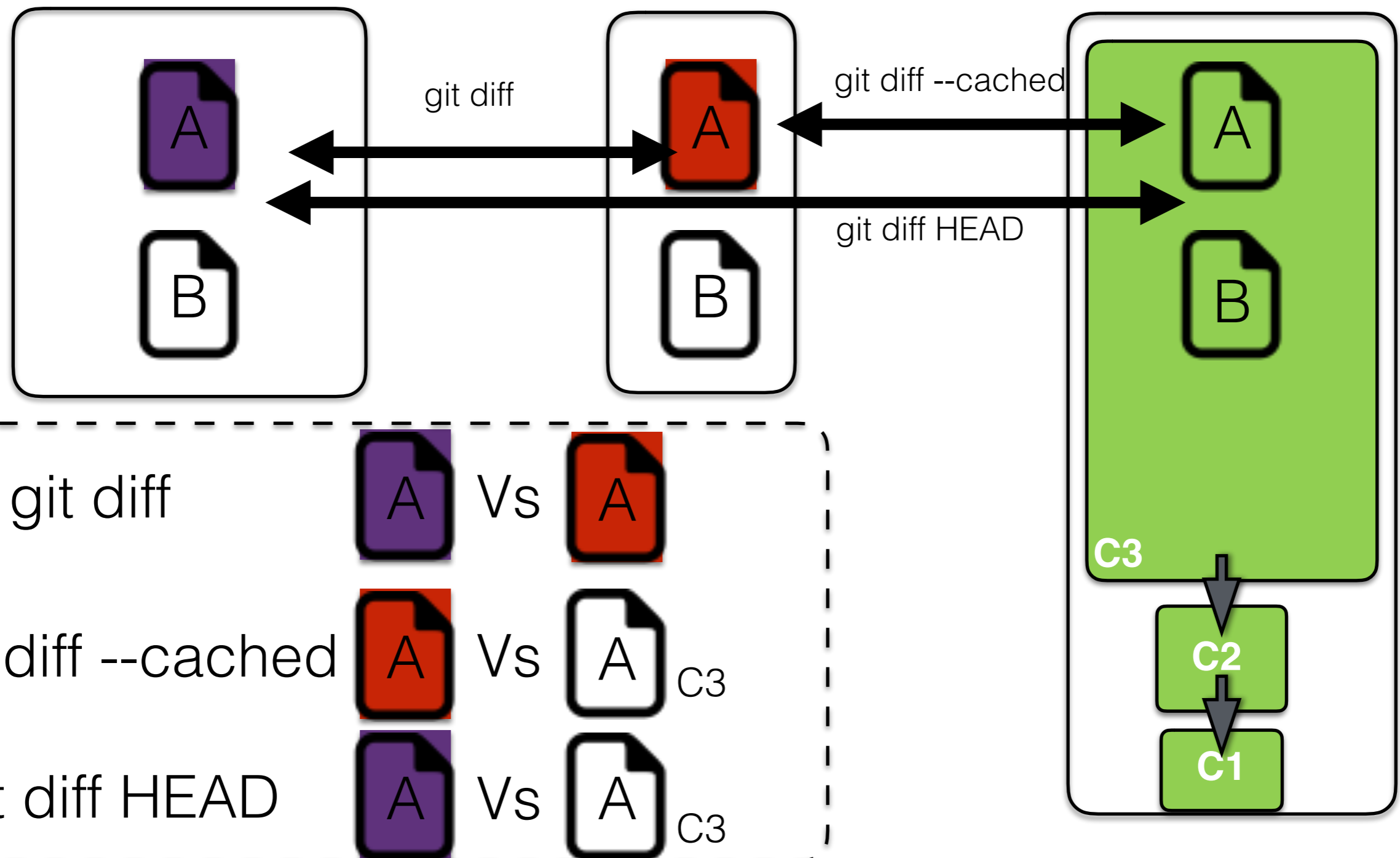
Demo #1

Git diff

Workspace

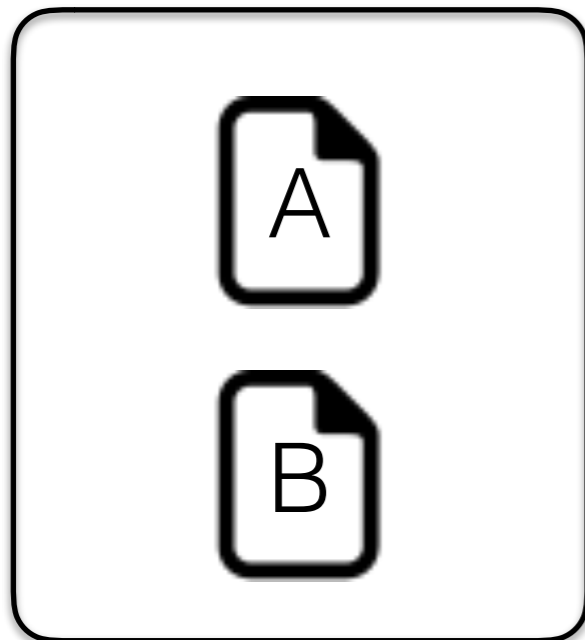
Index

Repository

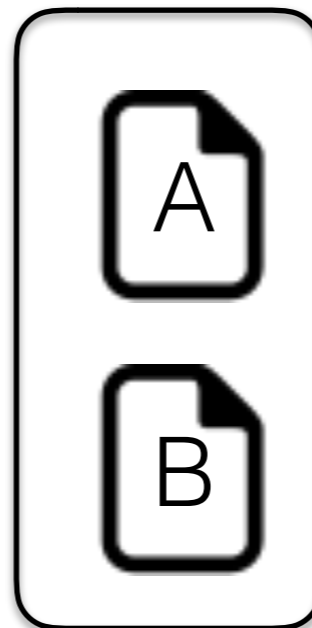


Git rm

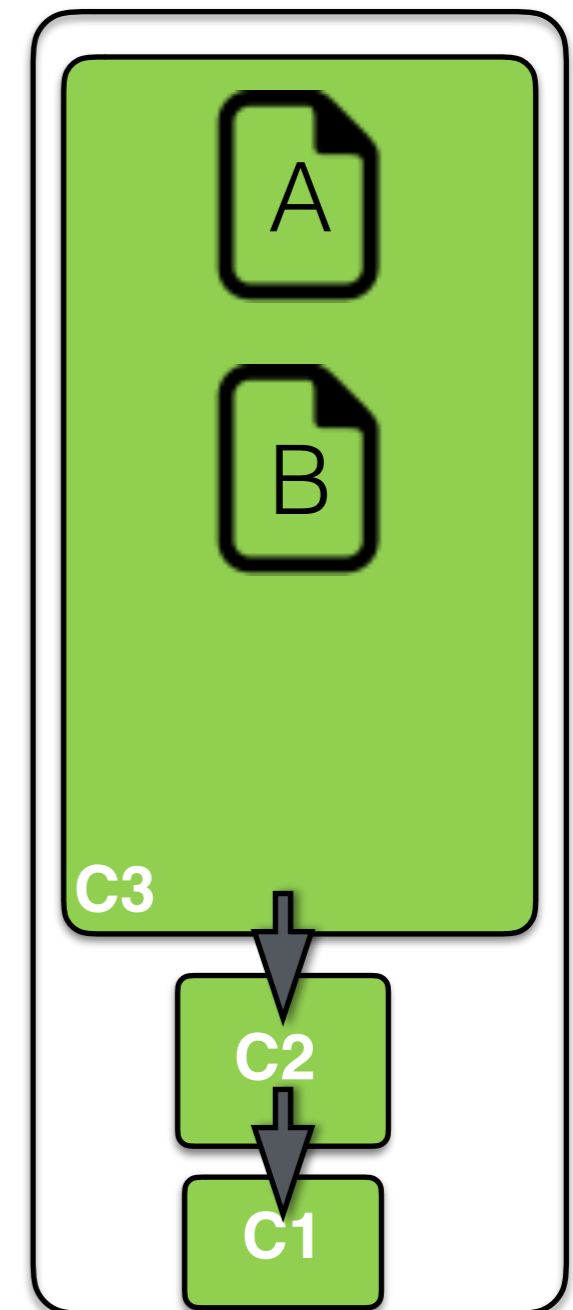
Workspace



Index



Repository

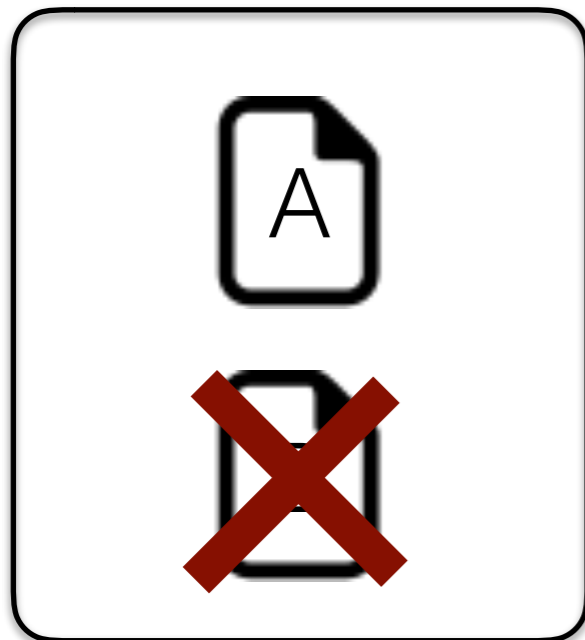


Action:

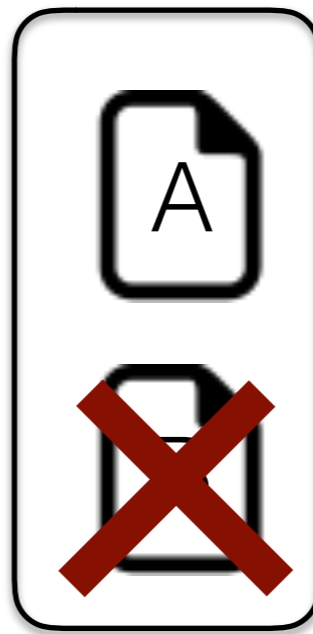
Git rm B

Git rm

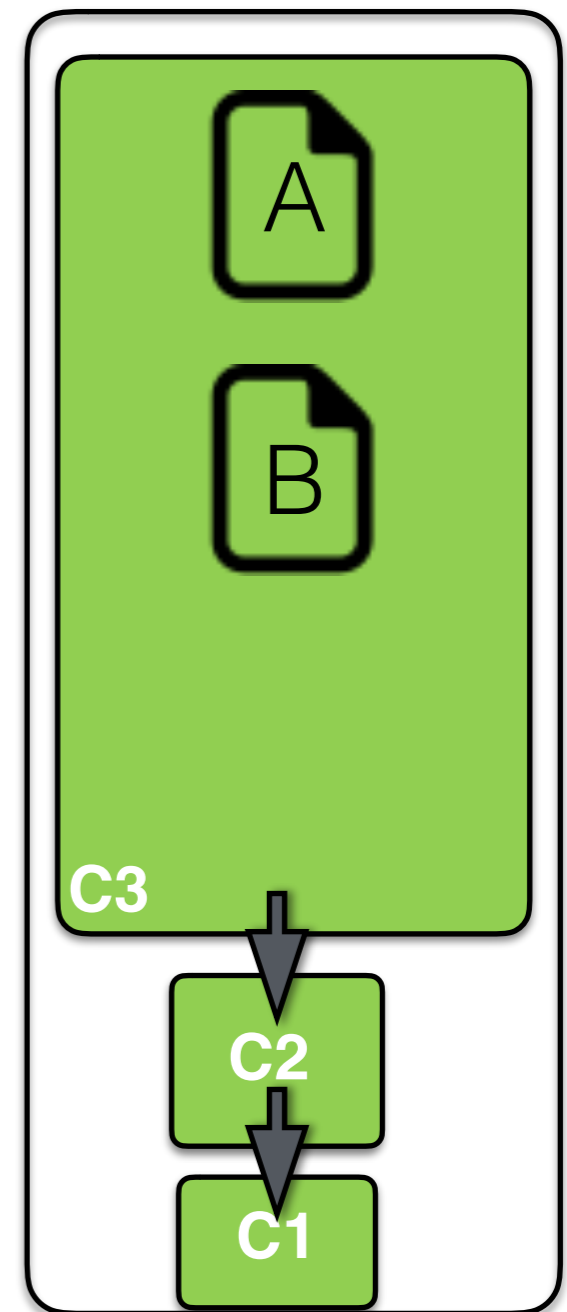
Workspace



Index



Repository



Action:

Git rm B

+> remove both in workspace

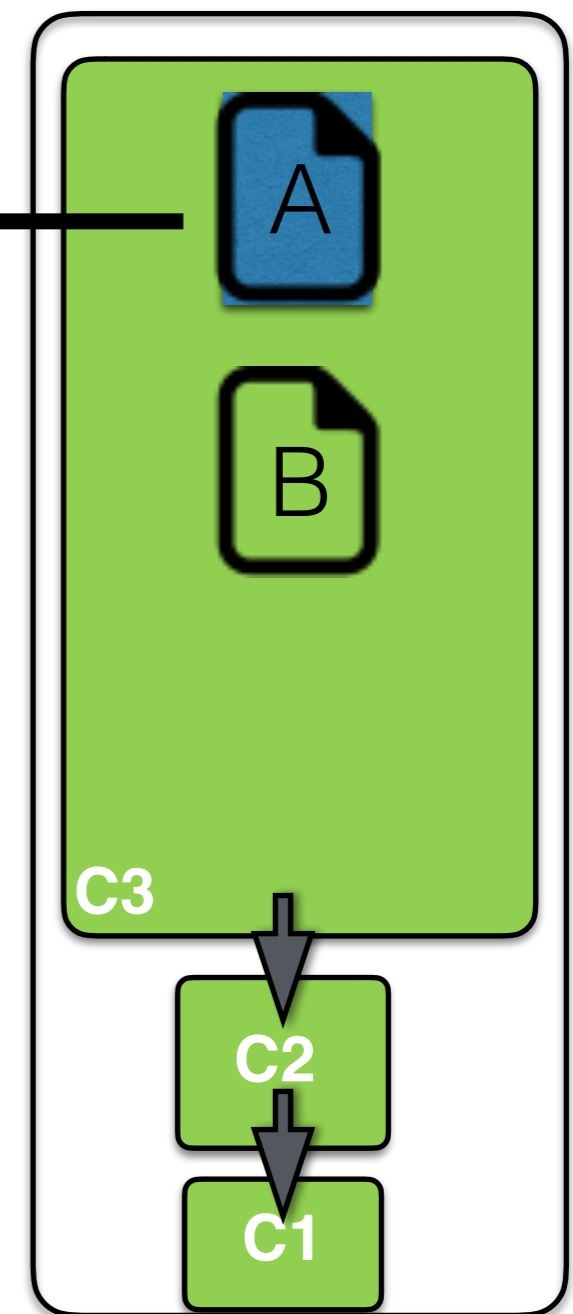
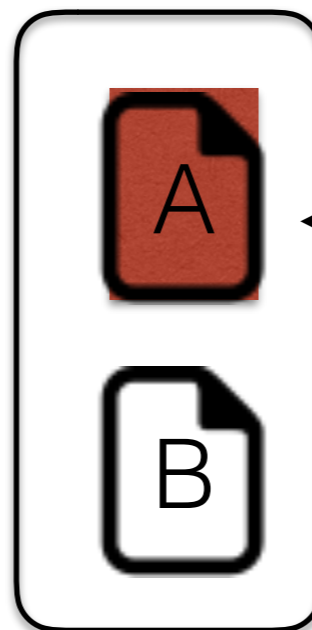
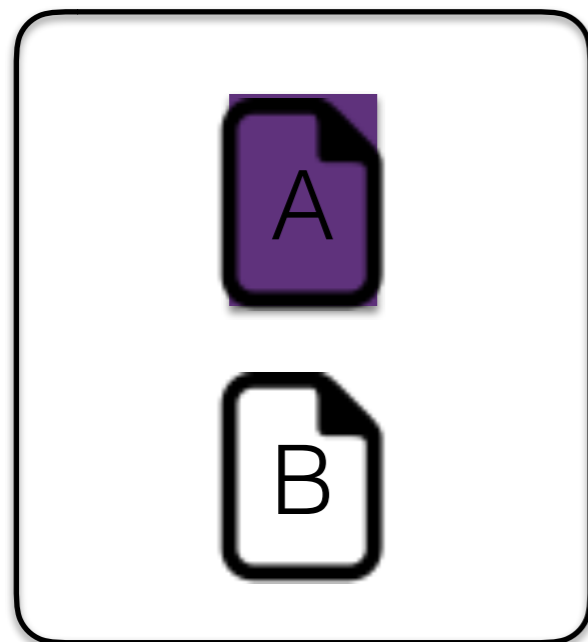
And remove in index (staging area)

Git reset

Workspace

Index

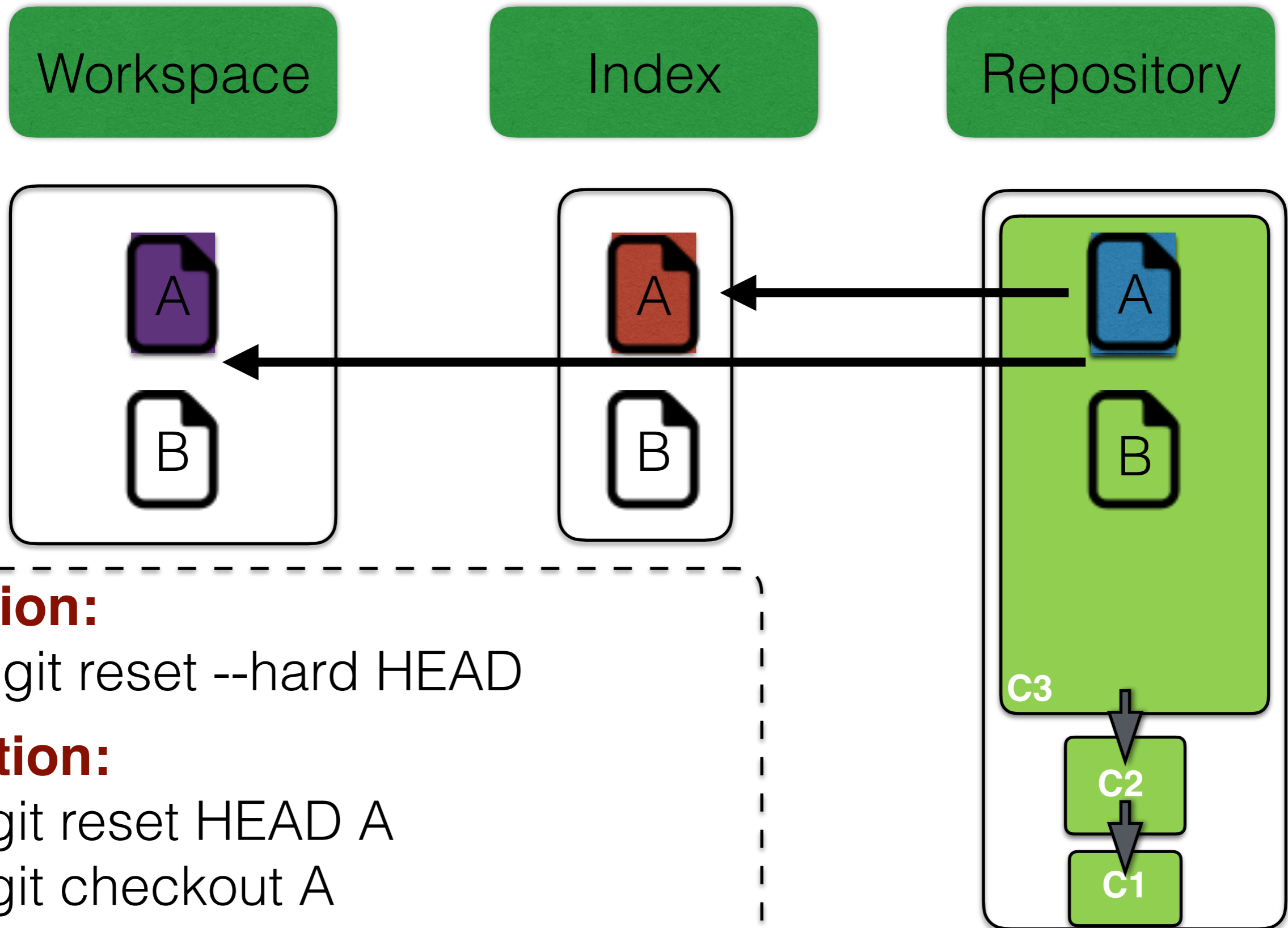
Repository



Action:

git reset HEAD A

Git reset



Restore file

Workspace

Index

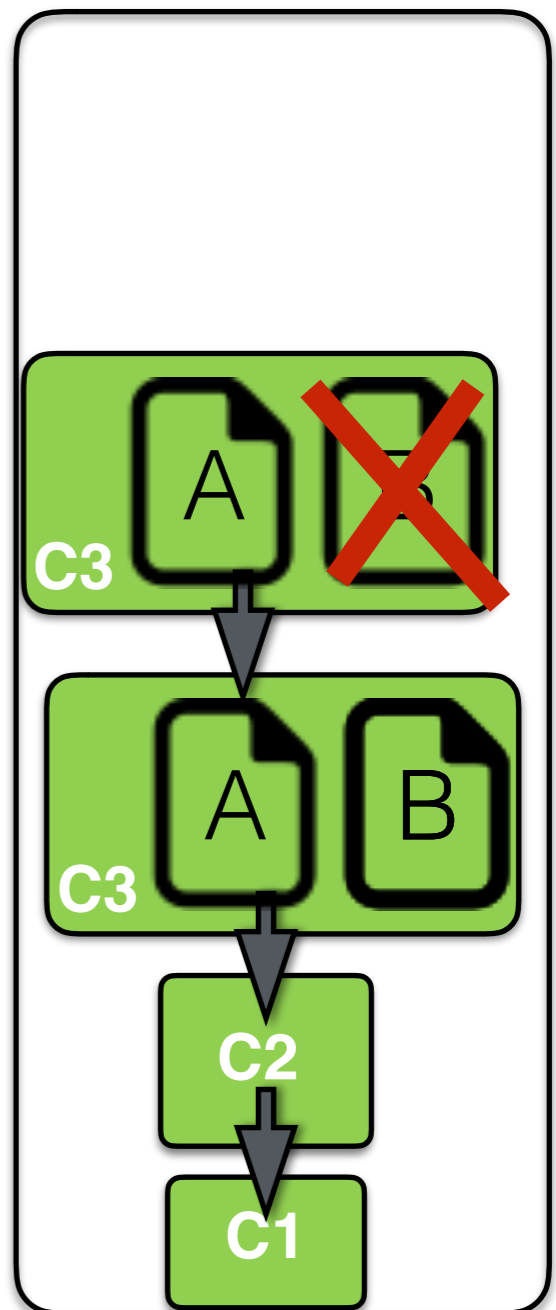
Repository



Action:

Git checkout C3 -- B

-> restore file B from version C3



Local project

Exercise #1

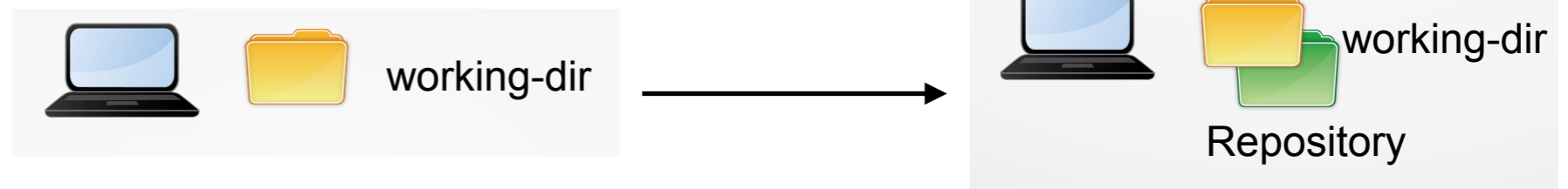
Starting with git

```
$ git config --global user.name "John Doe"  
$ git config --global user.email johndoe@example.com
```

.config/git/ignore, .gitignore

```
# Backup files left behind by the Emacs and vim editor.  
*~  
# Temporary files used by the vim editor.  
*.swp  
# compiled objects  
*.pyc  
*.o  
# directory fileter example (case sensitive)  
# ignore log dir  
Logs/
```

```
$ git init
```





single user/project

```
$ vim test.c  
$ vim test.h  
$ git status  
On branch master
```

Initial commit

Untracked files:
(use "git add <file>..." to include in what will be committed)

```
test.c  
test.h
```

nothing added to commit but untracked files present (use "git add" to track)



working-dir
Repository

adding file (for next commit)

```
$ git add test.c  
$ git status  
On branch master
```

Initial commit

Changes to be committed:
(use "git rm --cached <file>..." to unstage)

new file: test.c

Untracked files:
(use "git add <file>..." to include in what will be committed)

test.h



Commit

```
$ git commit -m'Add test.c'
[master (root-commit) 46ef322] Add test.c
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 test.c
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    test.h

nothing added to commit but untracked files present (use "git add" to track)
```



Repository

working-dir

checking modif

```
$ vim test.c
$ git diff
diff --git a/test.c b/test.c
index 0197793..0c7f097 100644
--- a/test.c
+++ b/test.c
@@ -1,4 +1,4 @@
int main()
{
-   int a=5;
+   int a=6;
}
$
```



Do it yourself

- install git
- configure the tools (name + email)
- create a local repository
 - commit one file then modify it and re-commit
- check “diff”, “log”, “status” functionality

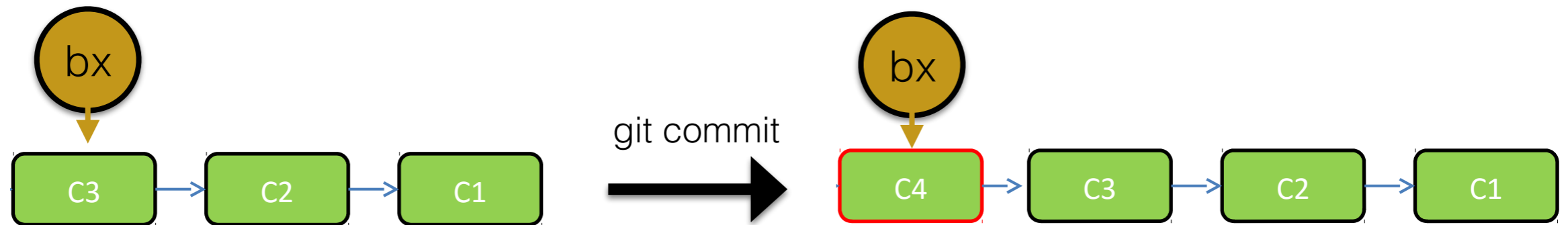
Workflow

branch in git

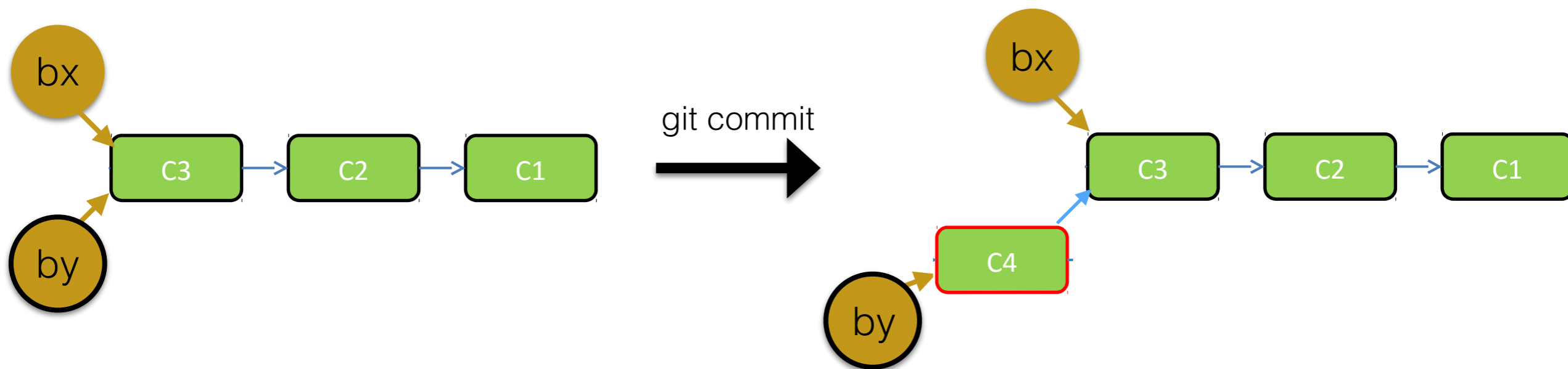
- Branch is **pointer** to a commit
- A branch can point at other commit, it can move!
- A branch is a way to organize your work and working histories
- Since commit know which commits they are based on, branch represents a commit and what came before it
- a branch is **cheap**, you can have multiple branch in the same repository and switch your working dir from one branch state to another

branches

- default branch: master
- When doing a commit, the branch moves to the new commit



- creating a new branch: add a pointer (git checkout -b by)
- only selected branch affected by commit!



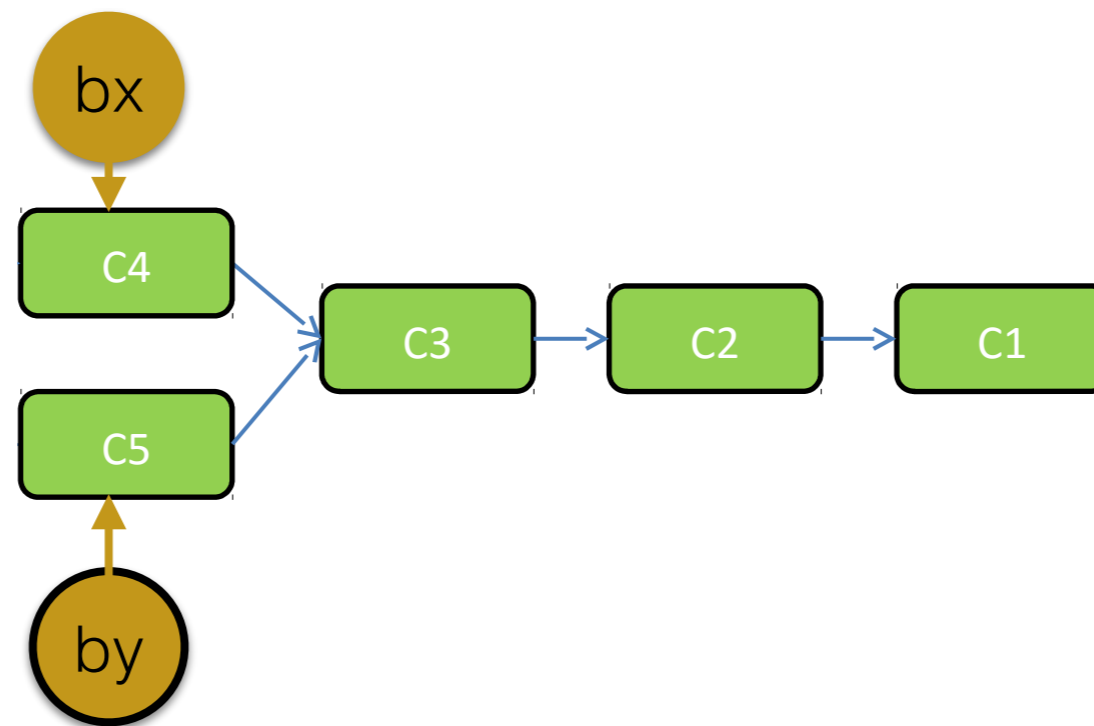
branches

create a new branch	<code>git checkout -b bx</code>
switch to a branch	<code>git checkout bx</code>
delete a branch	<code>git branch -d bx</code>
rename a branch	<code>git branch -m bx</code>
move a branch	<code>git branch -f bx rev</code>

- **master** : default created branch
- branch is cheap -> do it often
- branch allow to have short/long term parallel development (more on workflow later)

merging

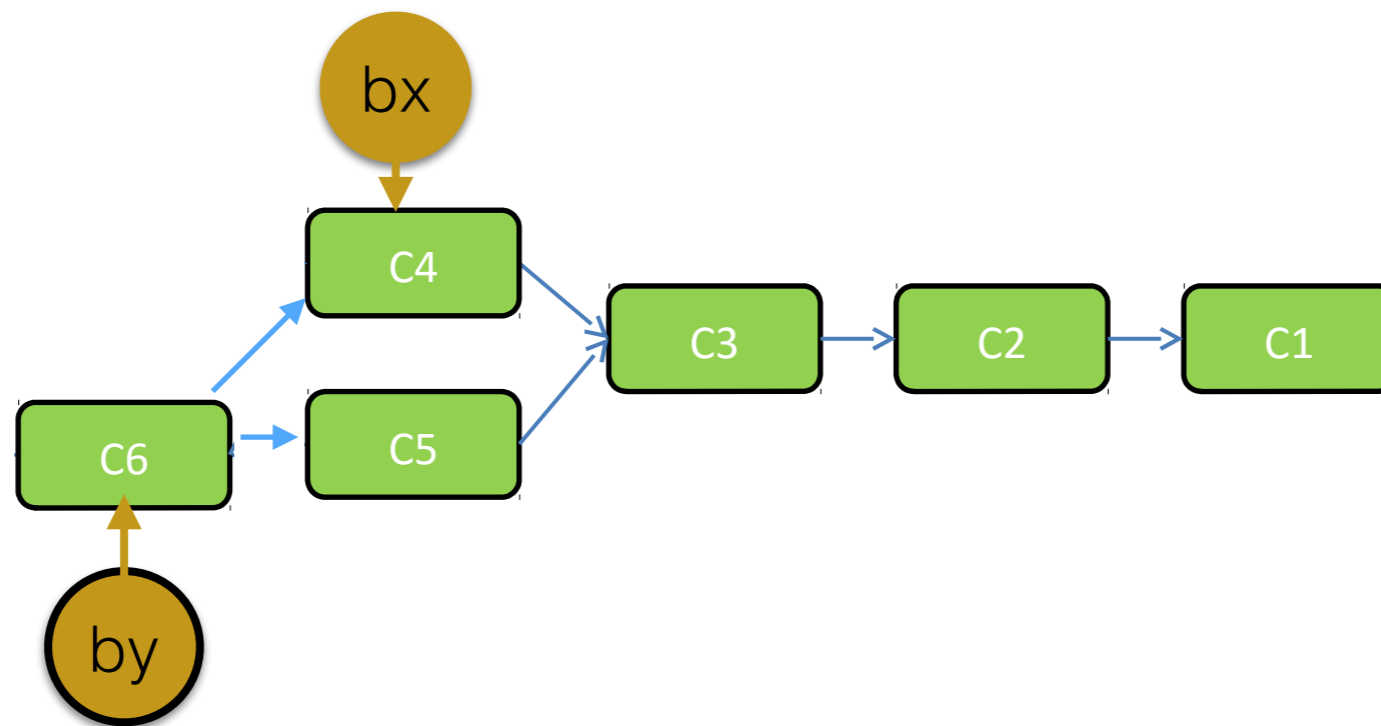
- The interest of branch is that you can **merge** them
- Include in one (branch) file the modification done somewhere else



git merge bx

merging

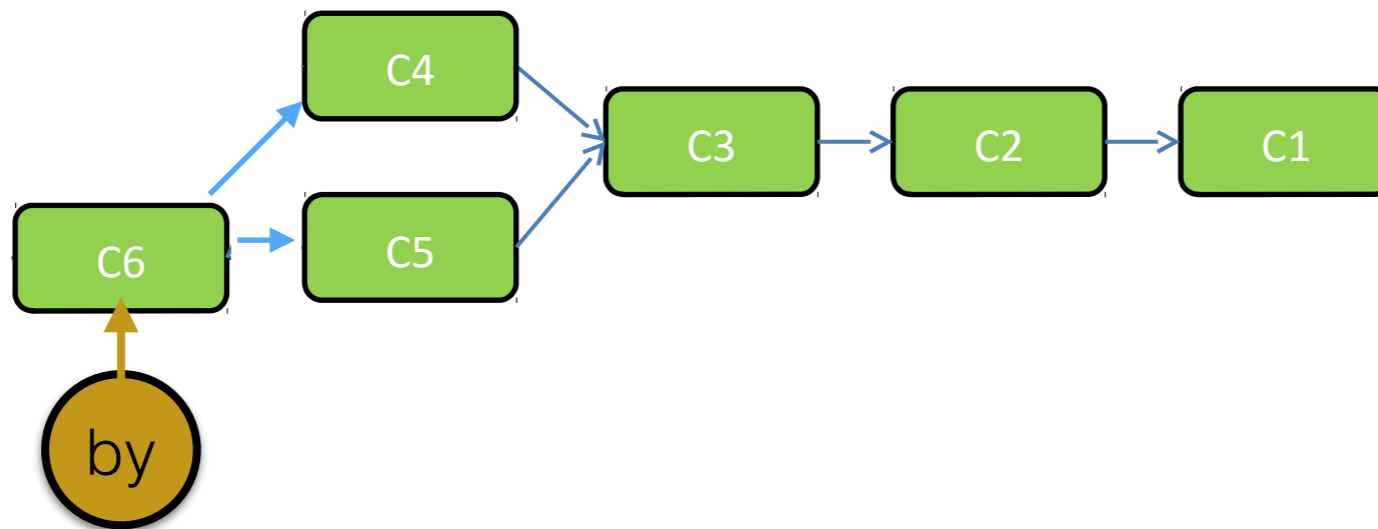
- merging two different modifications



git merge bx
git branch -d bx

merging

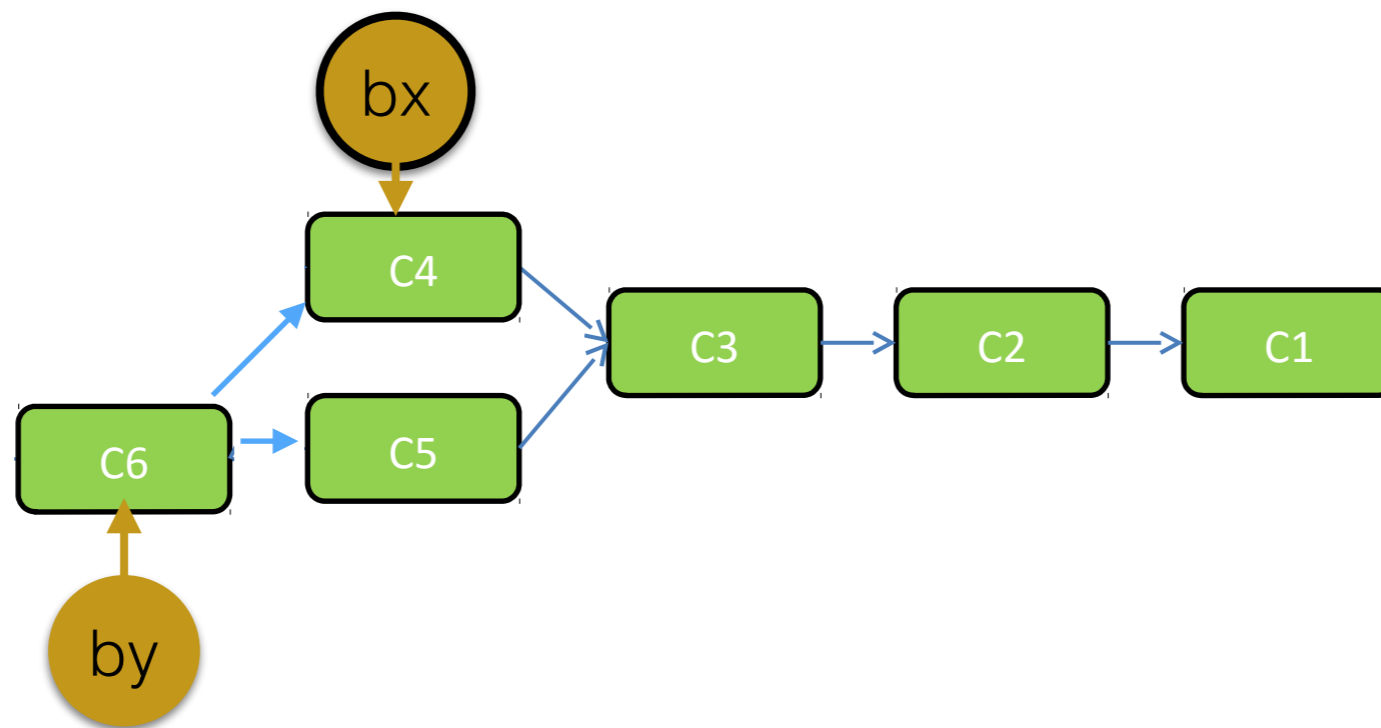
- merging two different modifications



git merge bx
git branch -d bx

merging

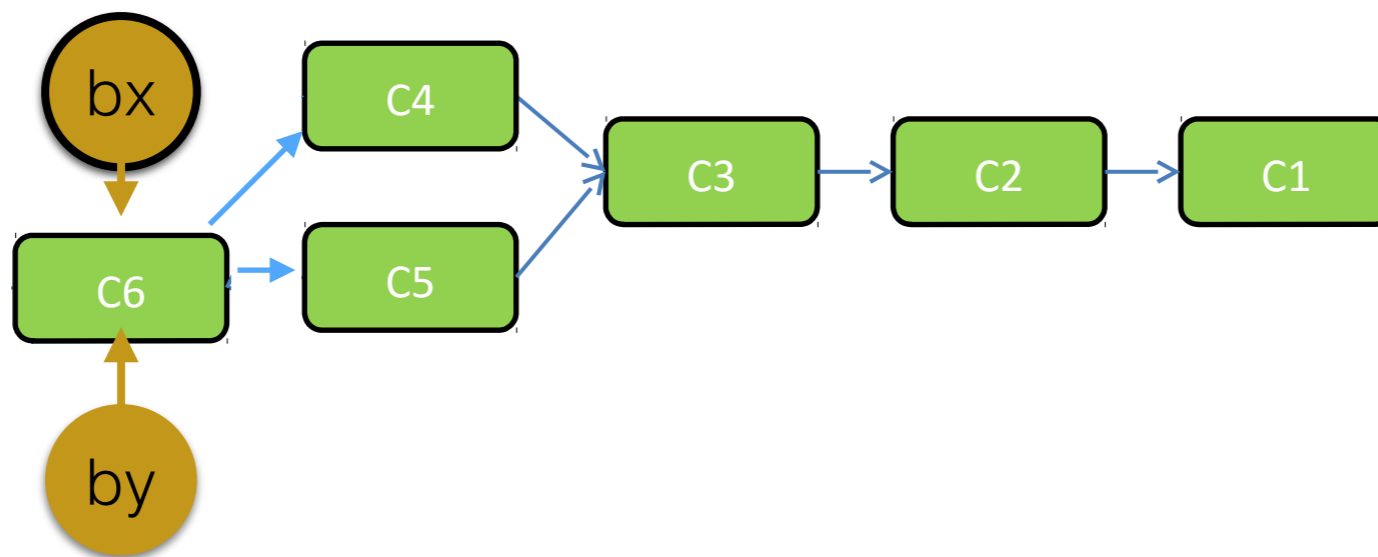
- merging two different modifications



git merge bx
git checkout bx
git merge by

merging

- merging two different modifications



git merge bx
git checkout bx
git merge by

merging can lead to conflict

```
[gittest]$ git merge hello  
Auto-merging helloworld.py  
CONFLICT (content): Merge conflict in helloworld.py  
Automatic merge failed; fix conflicts and then commit the result.  
[gittest]$ █
```

Conflict

- **Multiple version of files** are great
 - Not always easy to know how to merge them
 - Conflict will happen (same line modify by both user)
- Conflict need to be resolved manually!
 - Boring task
 - need to understand why a conflict is present!
- **Do not be afraid of conflict!** Do not try to avoid them at all cost!
- stay in sync as most as possible and keep line short

Conflict

```
print "Hello World"
<<<<<<< HEAD
print "changed from master branch"
=====
print "print from branch to be merged"
>>>>>>> hello
```

Edit the file to the “correct” version

```
print "Hello World"
print "print from master branch"
print "and from branch to be merged"
█
```

Run

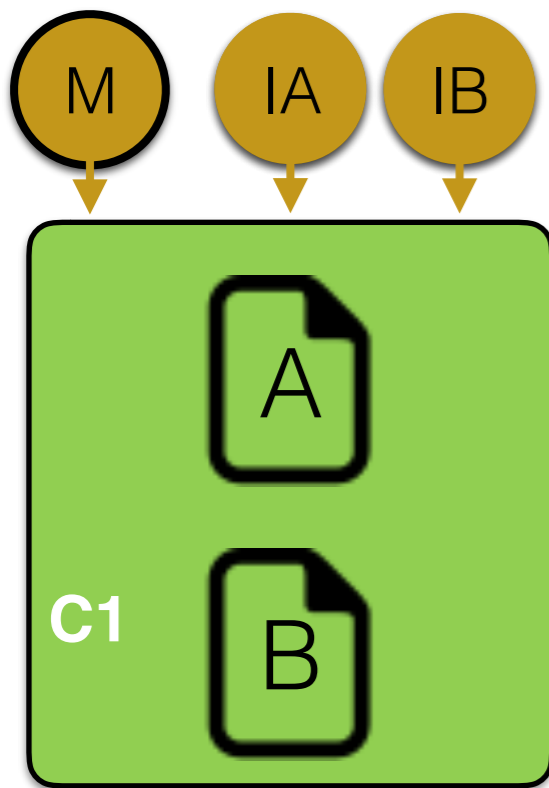
->git add .

-> git commit

Keep history clean: Rebase

- Instead of merging, replays set of changes on top of another branch
- Affects the “rebased” branch only
- Changes the history of commits
- Can be dangerous
- Very useful to remove history clutter
- Simple rule, use locally only

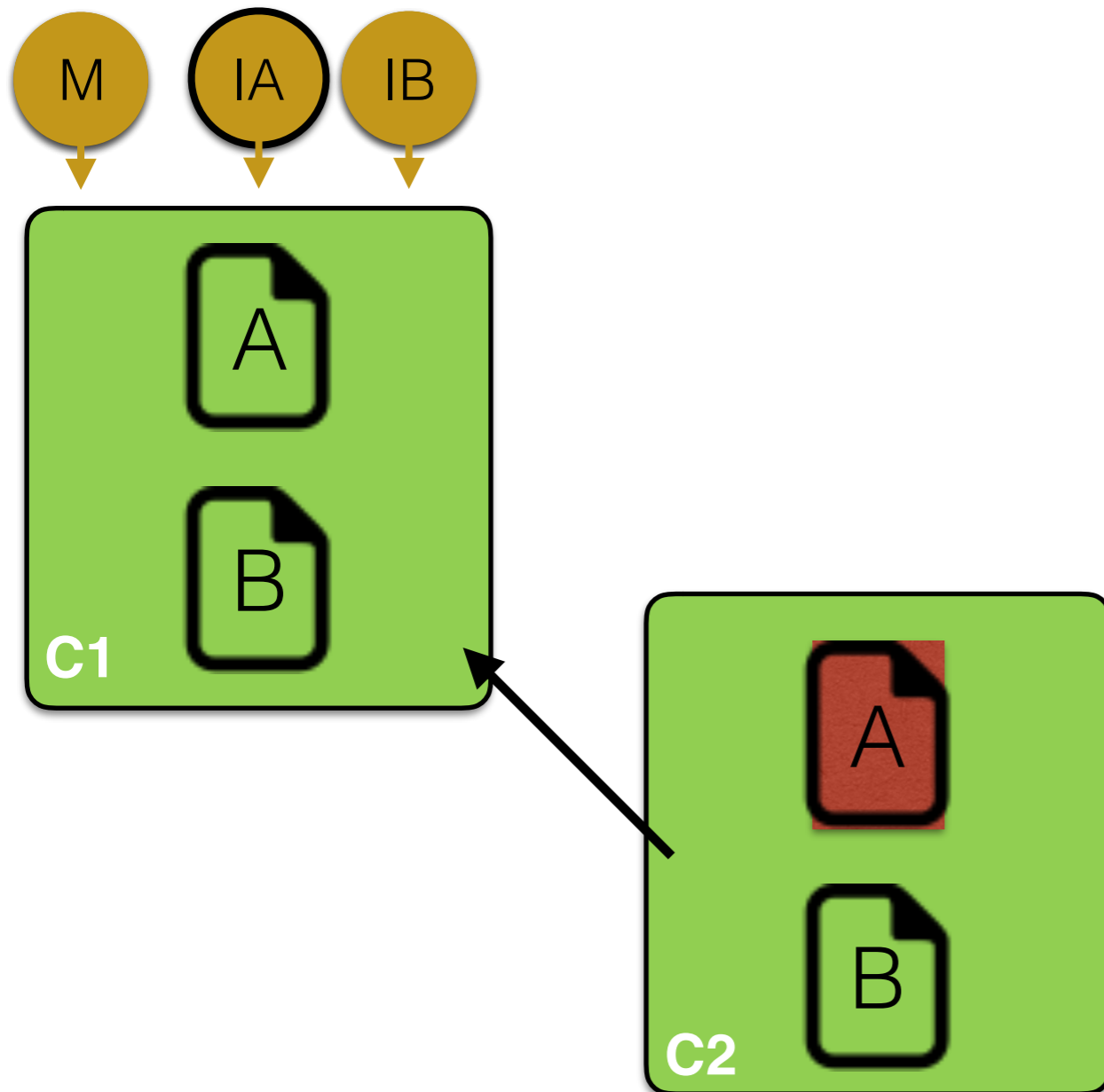
Keep history clean: Rebase



Keep history clean: Rebase

Git checkout IA

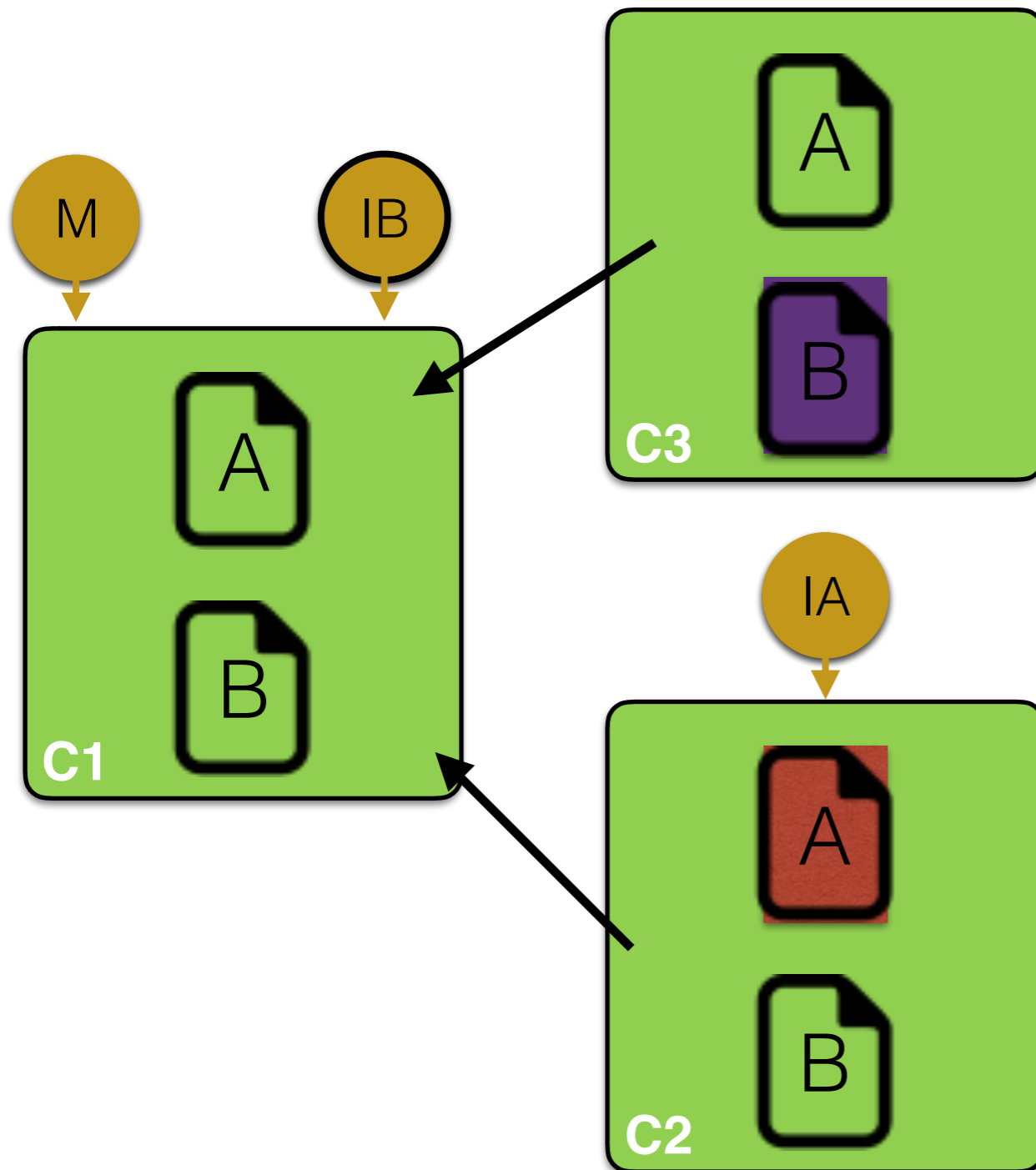
Git commit



Keep history clean: Rebase

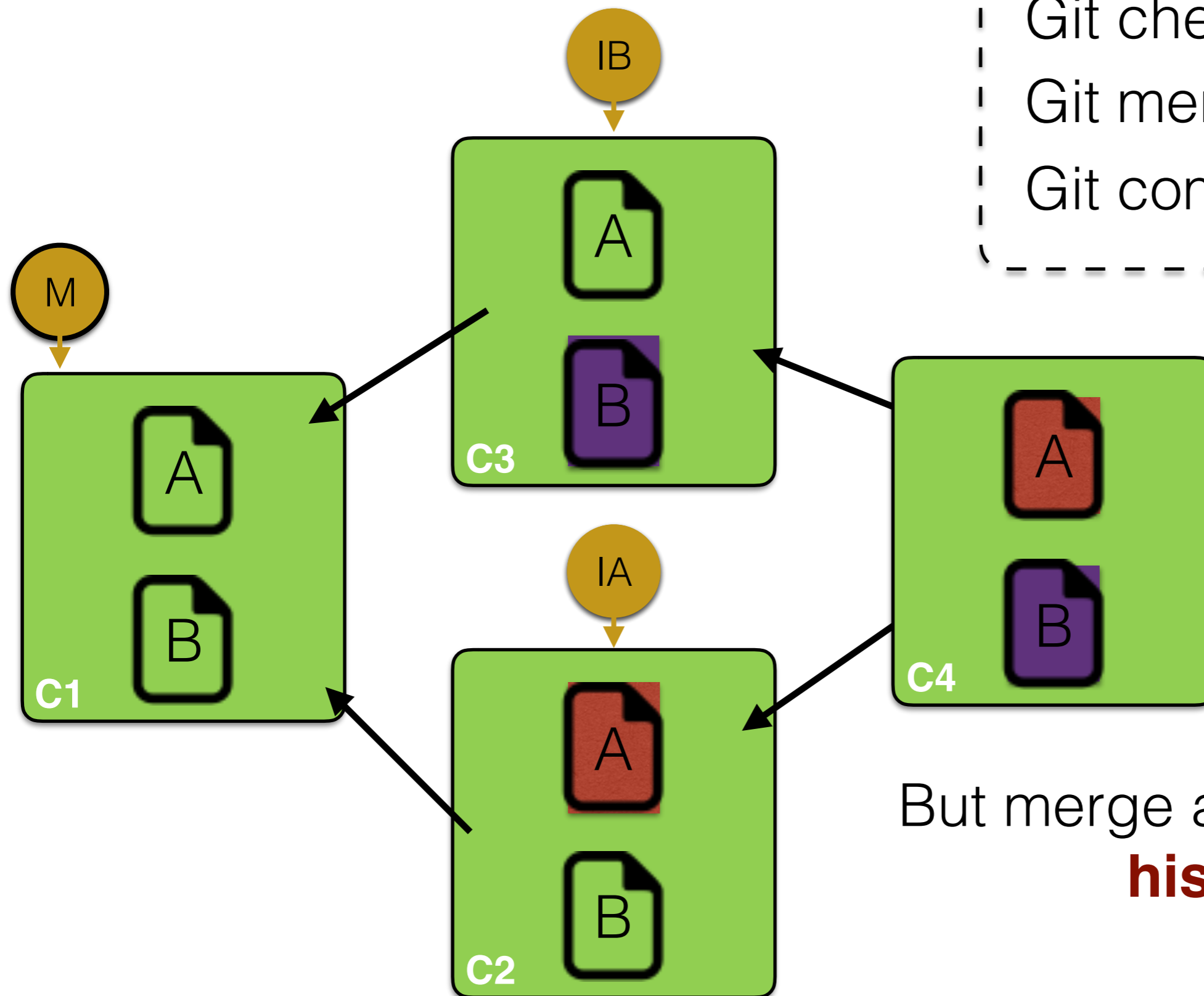
Git checkout IB

Git commit



I want to include **BOTH** changes in master branch

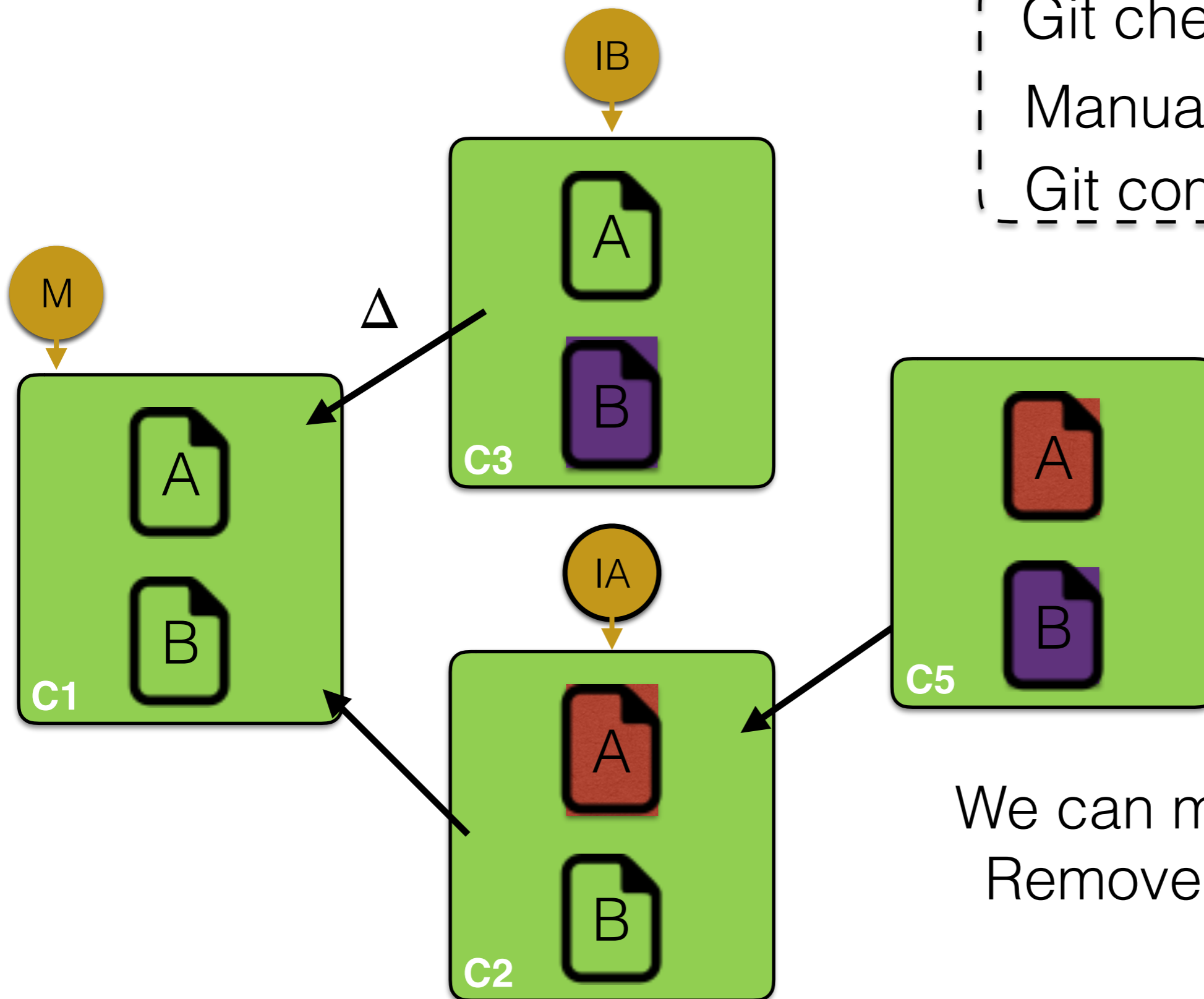
Keep history clean: Rebase



Git checkout M
Git merge IA IB
Git commit

But merge are **not clean history**

Keep history clean: Rebase



Git checkout IA
Manual change of B
Git commit

We can merge M (ff)
Remove IB and IA

Keep history clean: Rebase

This is **not easy** to do
-> let automate that
-> “rebase”

Git checkout IA

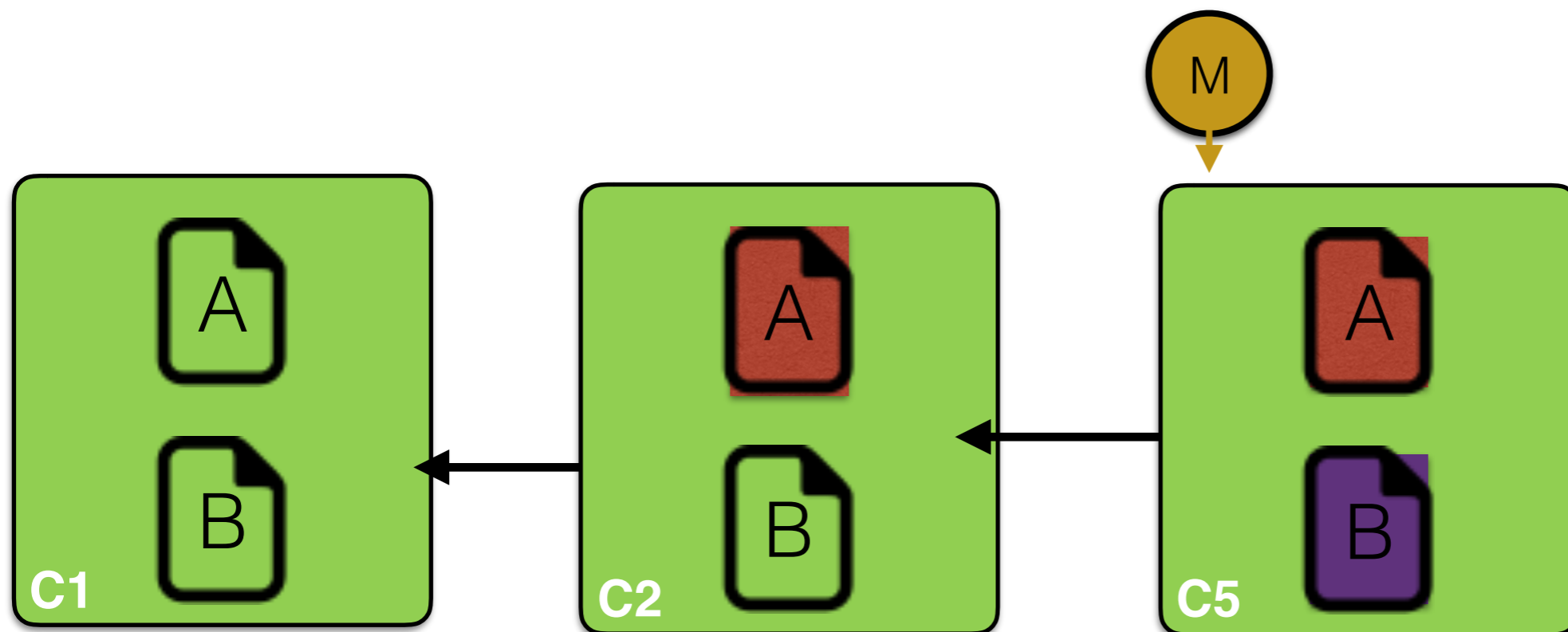
Manual change of B

Git commit

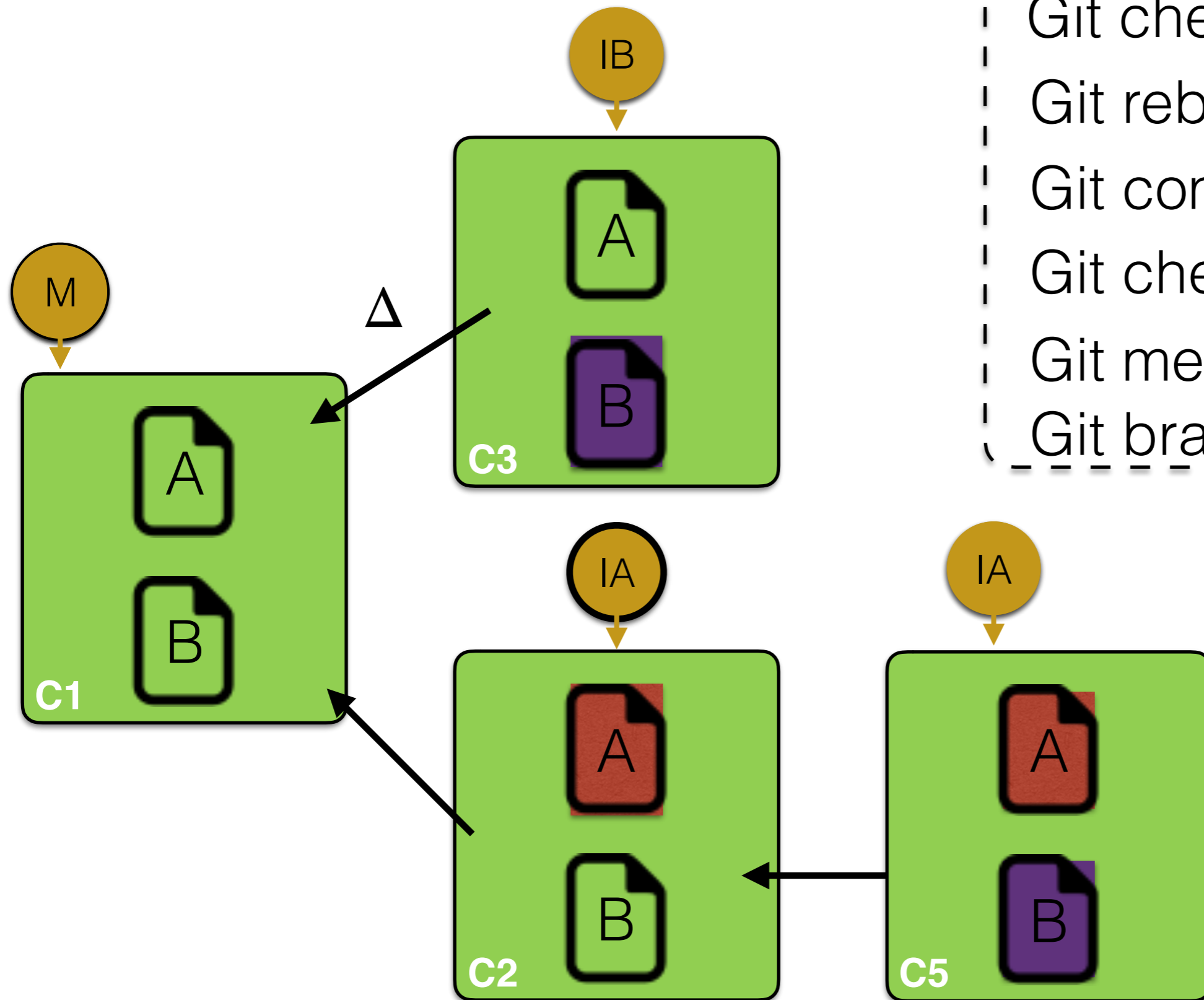
Git checkout M

Git merge IA

Git branch -d IA IB



Keep history clean: Rebase



Git checkout IA

Git rebase IB

Git commit

Git checkout M

Git merge IA

Git branch -d IA IB




working-dir
Repository


Do it yourself

- create two branch on your repository
- make new commit on each branch
- merge (test case with and without conflict)
- redo the same but use the rebase method




Team Work


GitHub/Gitlab



Search or jump to... 

[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)



  




Learn Git and GitHub without any code!


Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

[Read the guide](#) [Start a project](#)


 Our new Terms of Service and Privacy Statement are in effect. 

Repositories [New repository](#)

 [oliviermattelaer/singularity-recipe](#)


 [oliviermattelaer/MGISR-1](#)

Browse activity

 dcolignon starred oliviermattelaer/Singularity-Tutorial 7 days ago

oliviermattelaer/Singularity-Tutorial

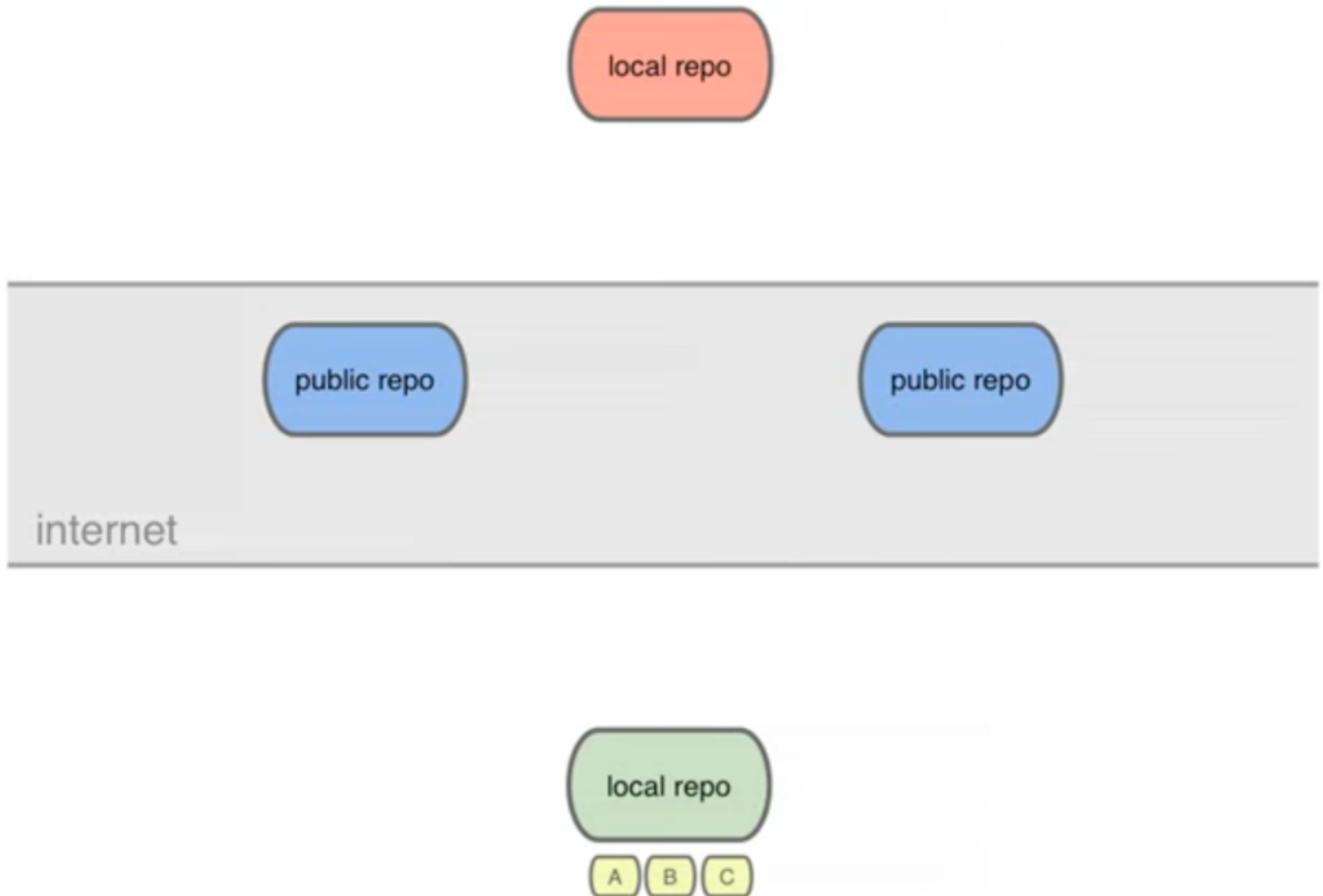
Materials for 3 hour hands-on workshop entitled "Creating and running software containers with Singularity"

 1 Updated Oct 31

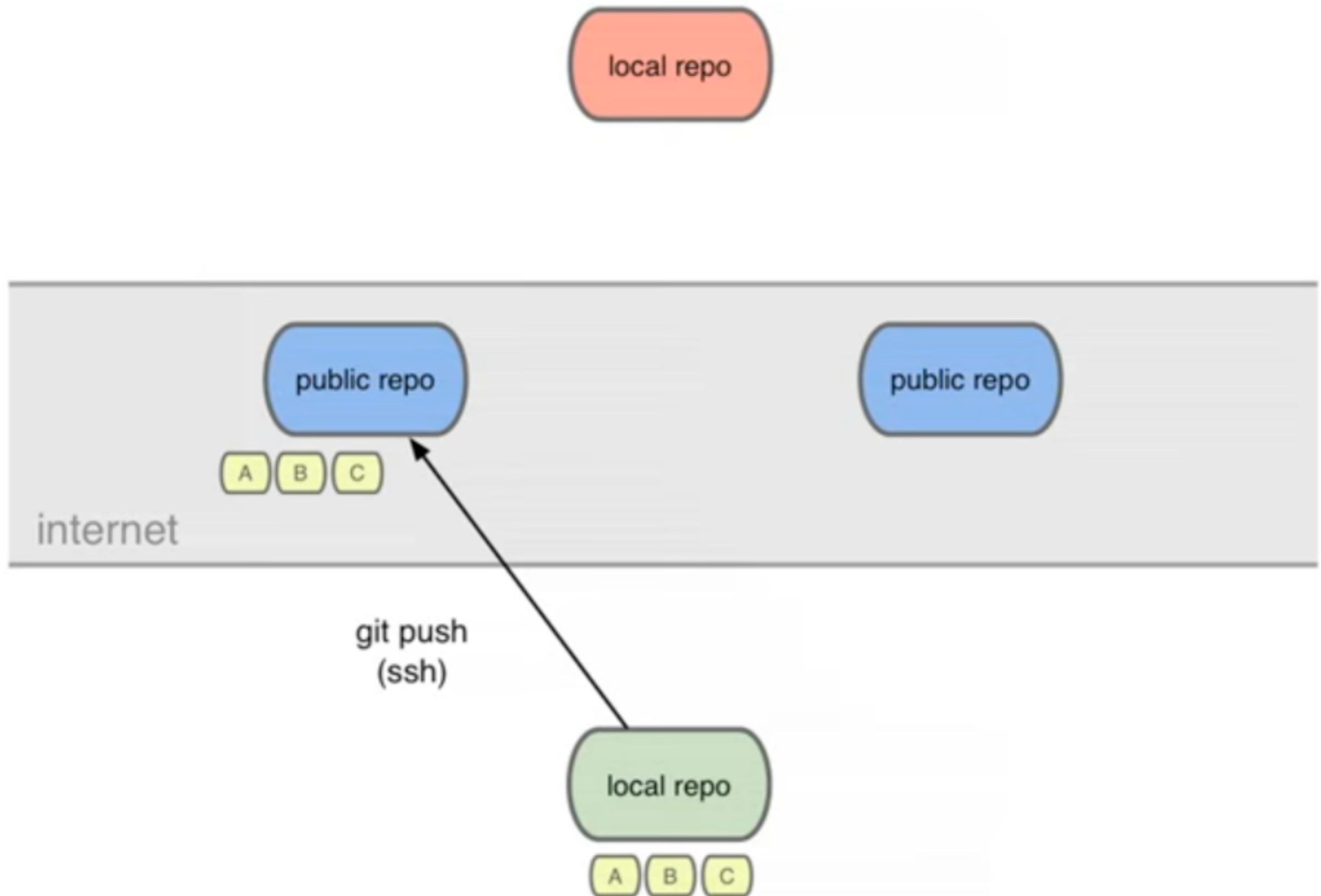
[★ Star](#)

[Discover repositories](#)

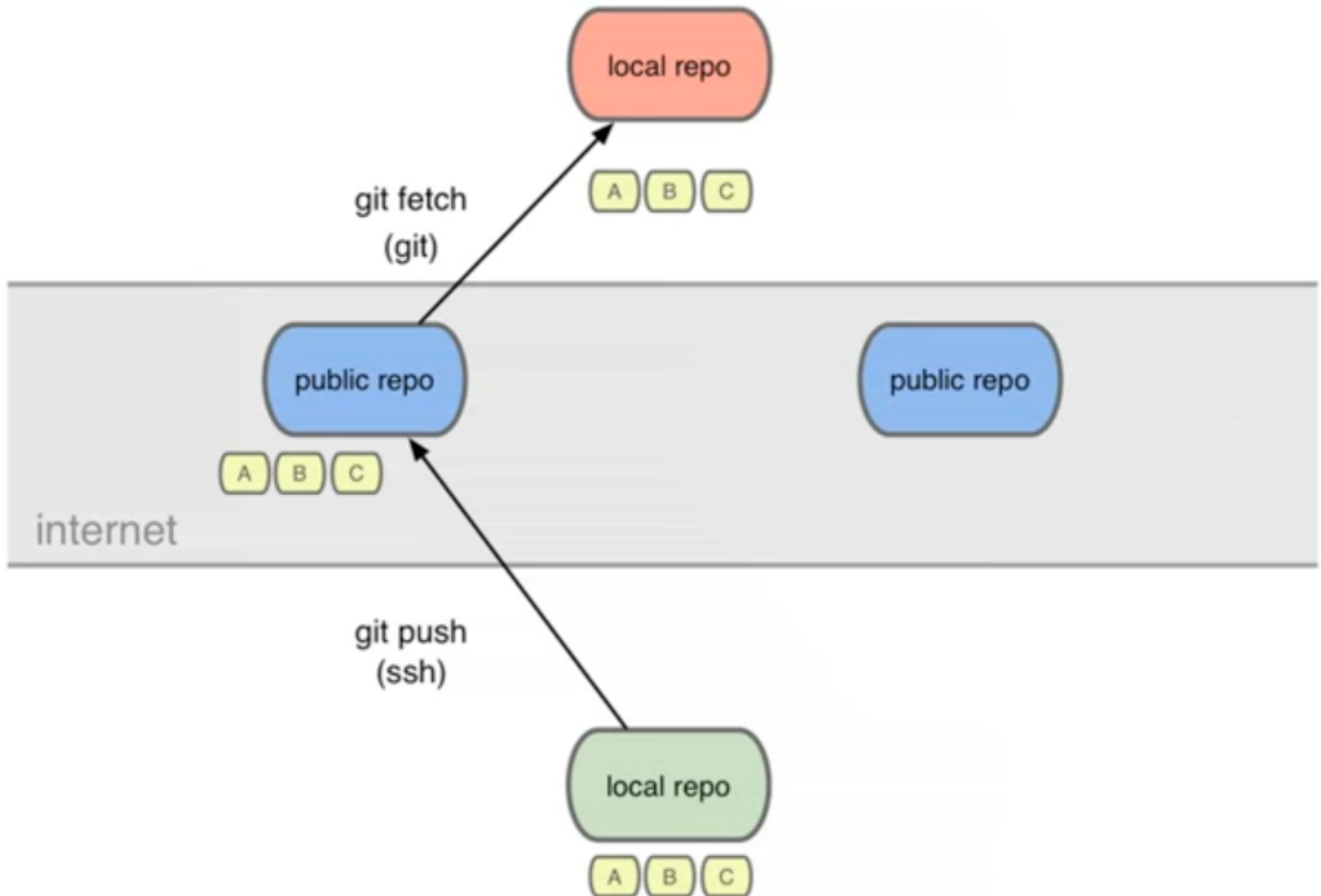
Collaboration



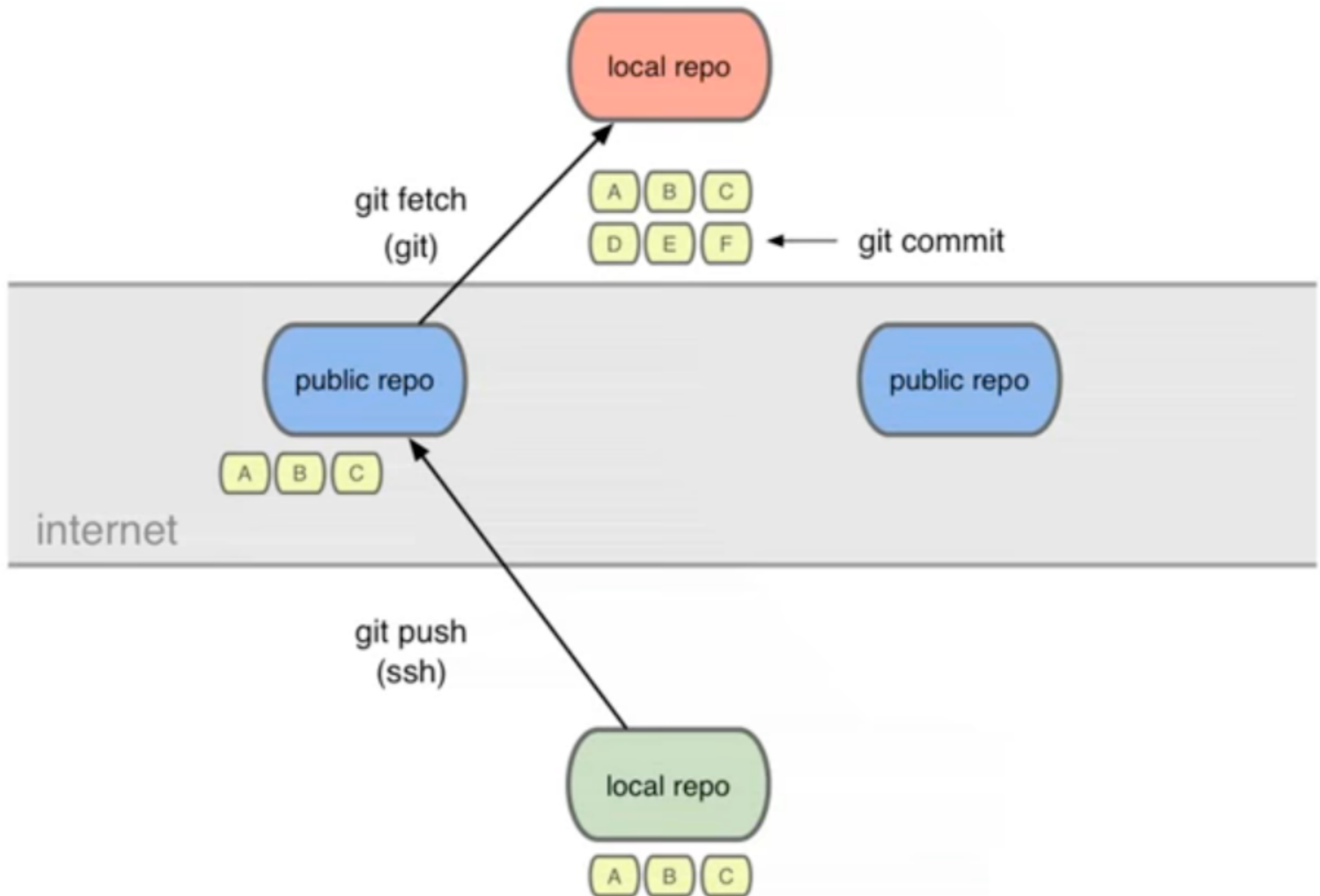
Collaboration



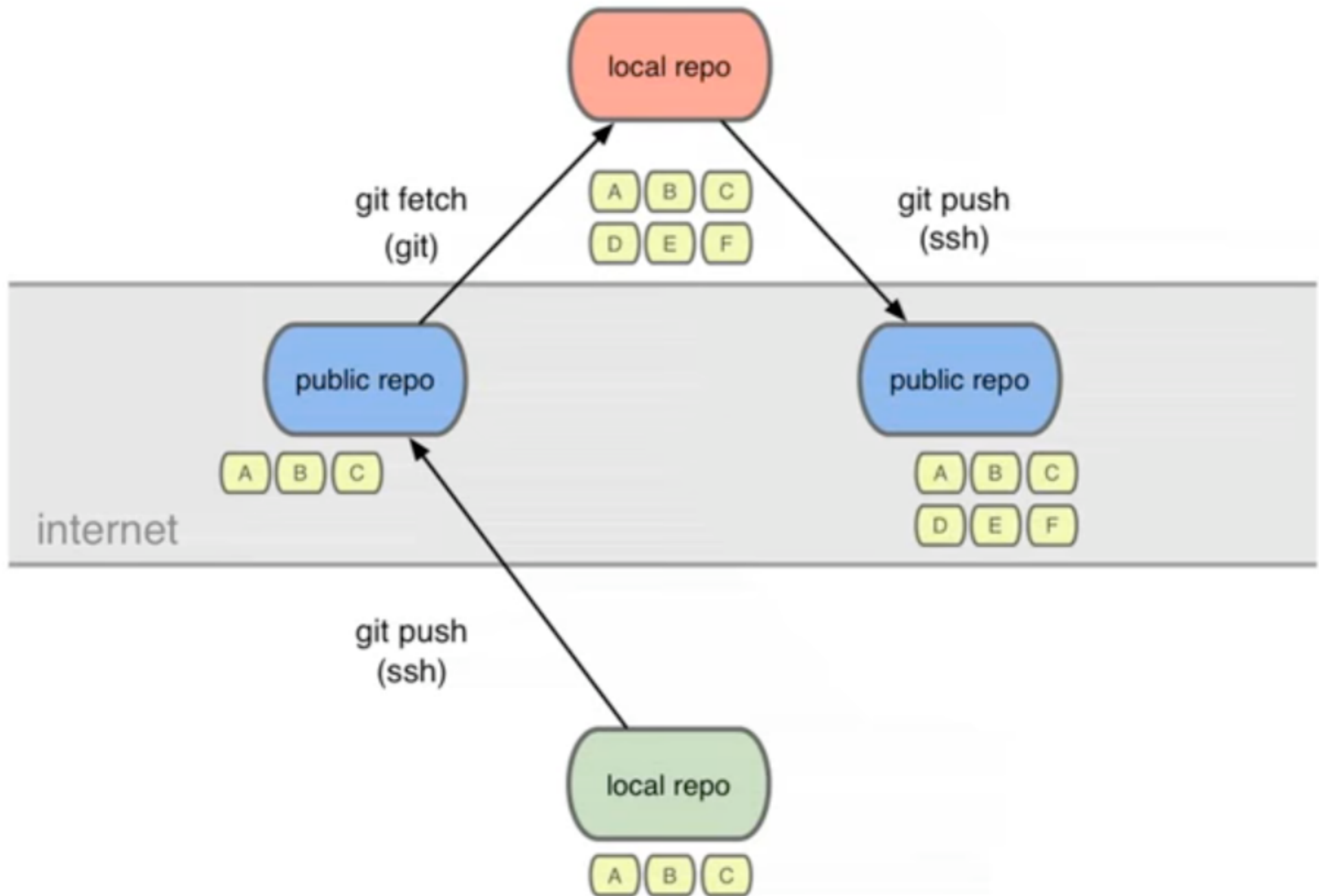
Collaboration



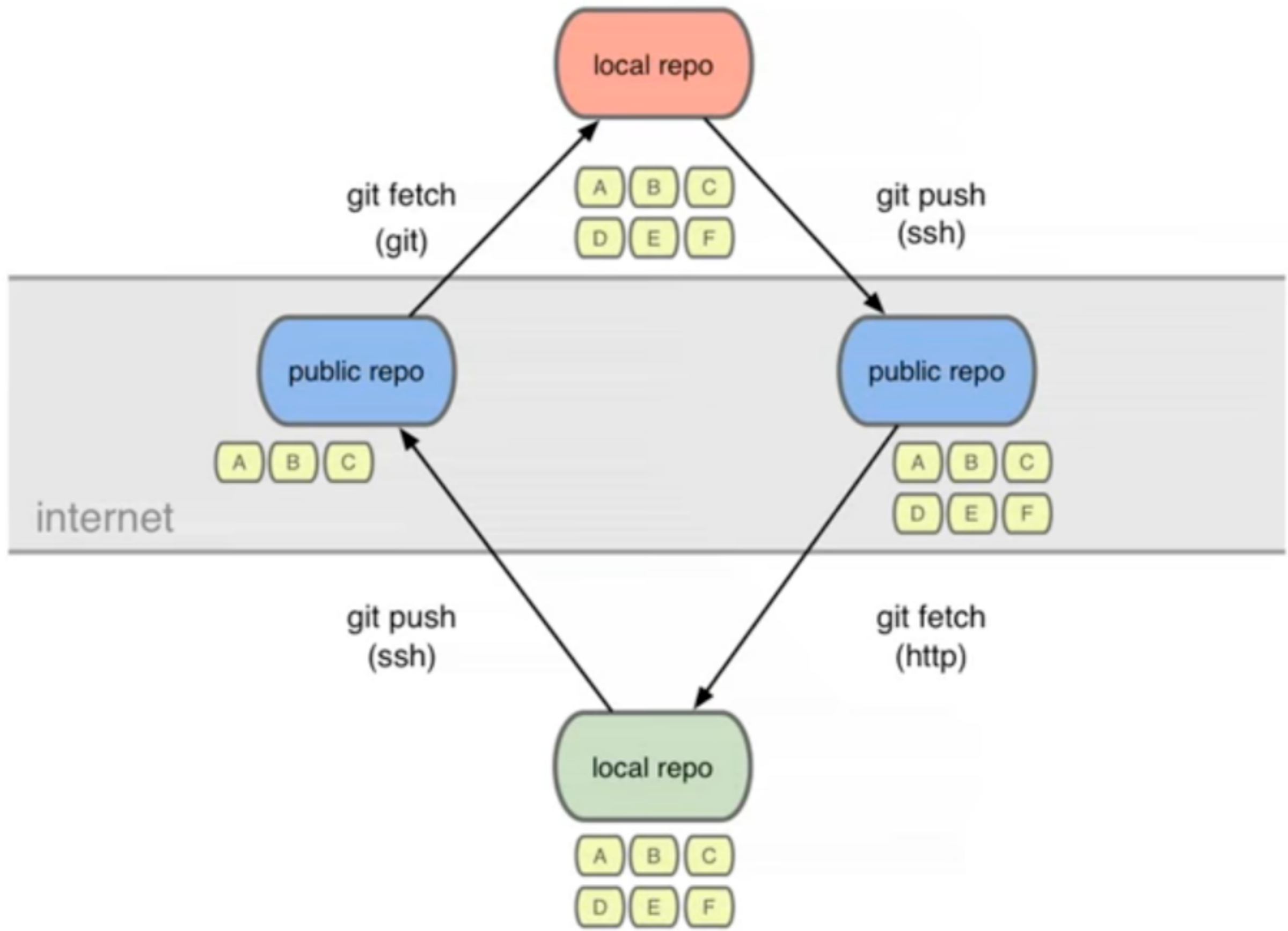
Collaboration



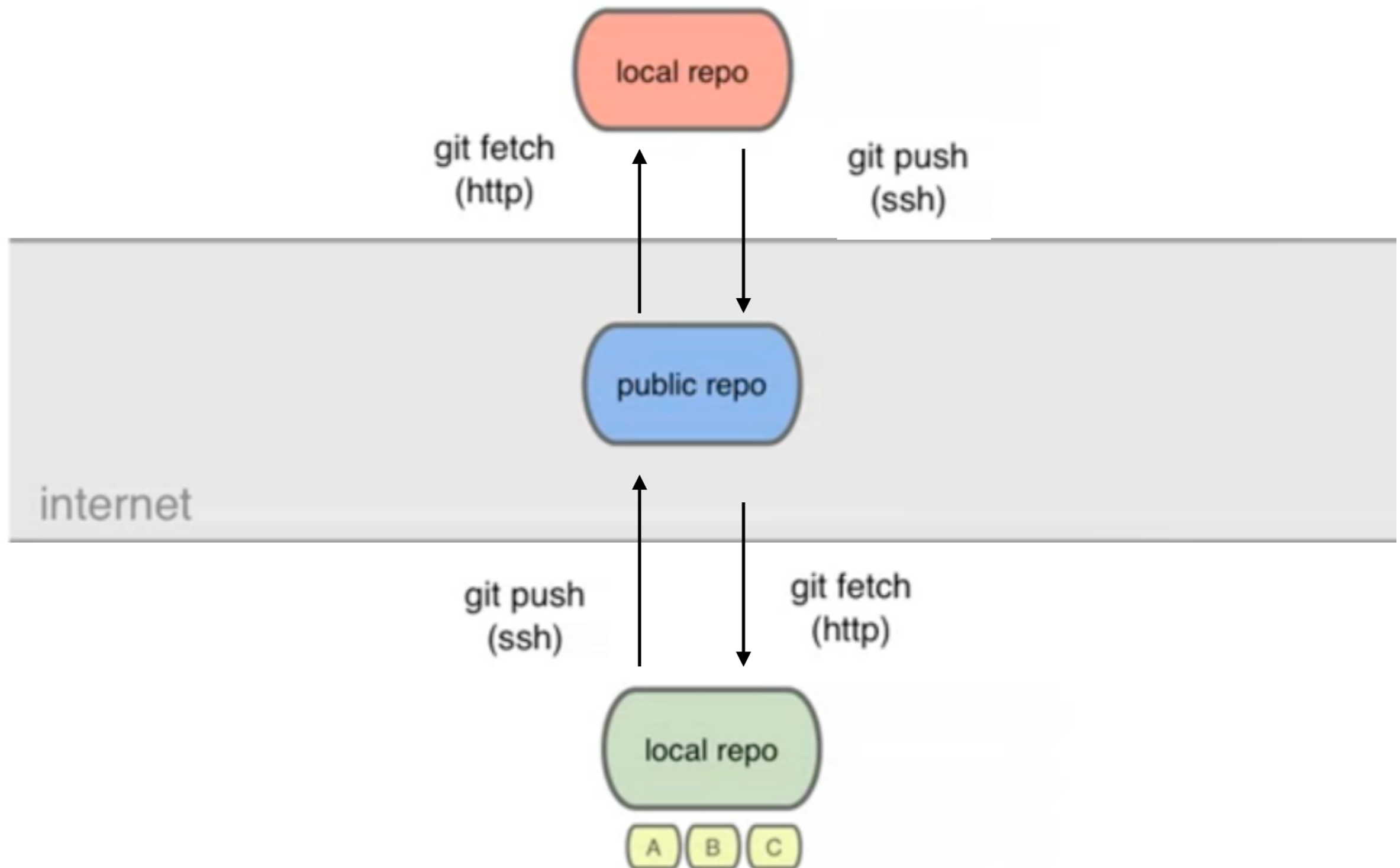
Collaboration



Collaboration



Collaboration (II)



Remote Branches

This is a remote branch

My Machine

origin/master

master

C1

C0

origin is just a name for a "remote"

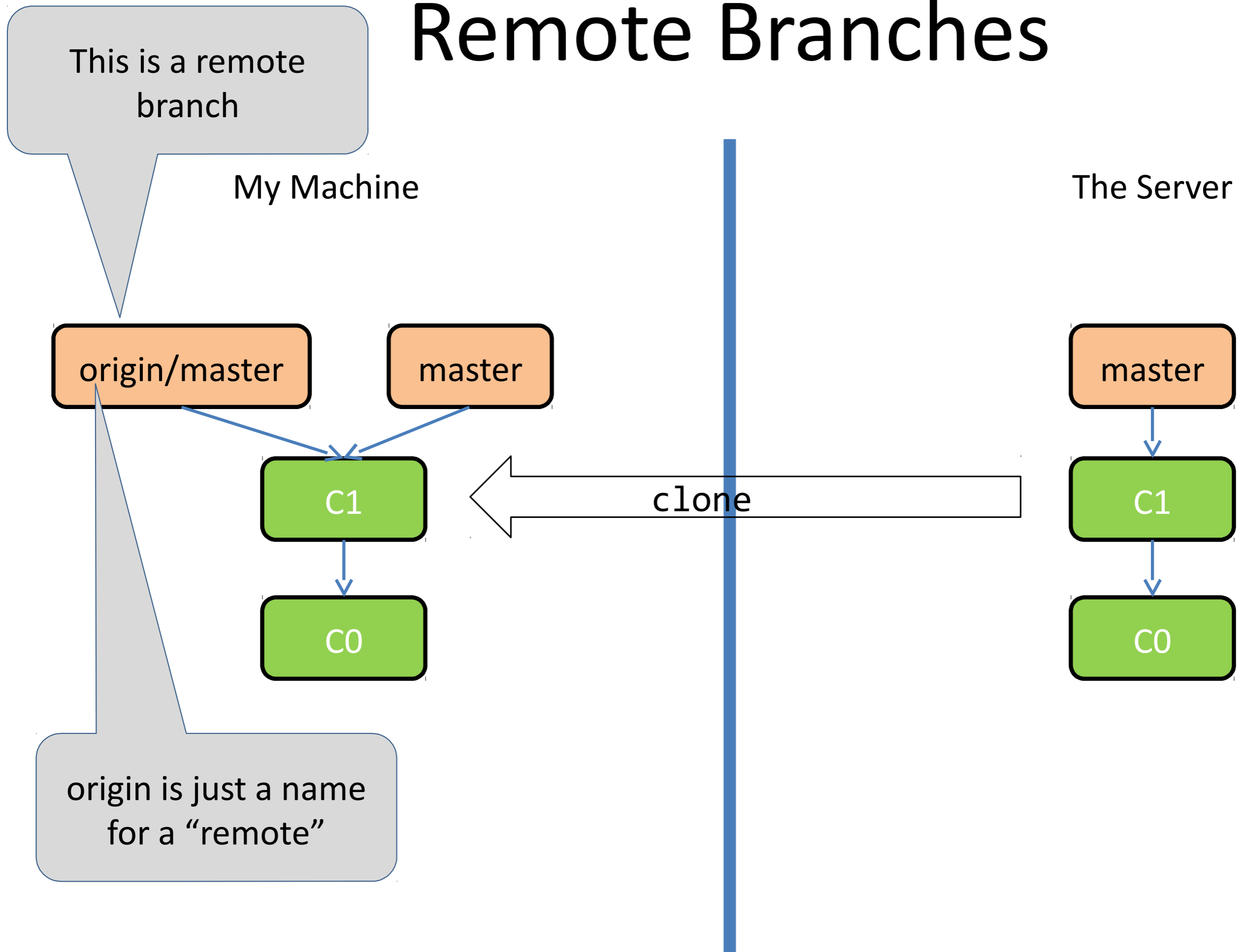
clone

The Server

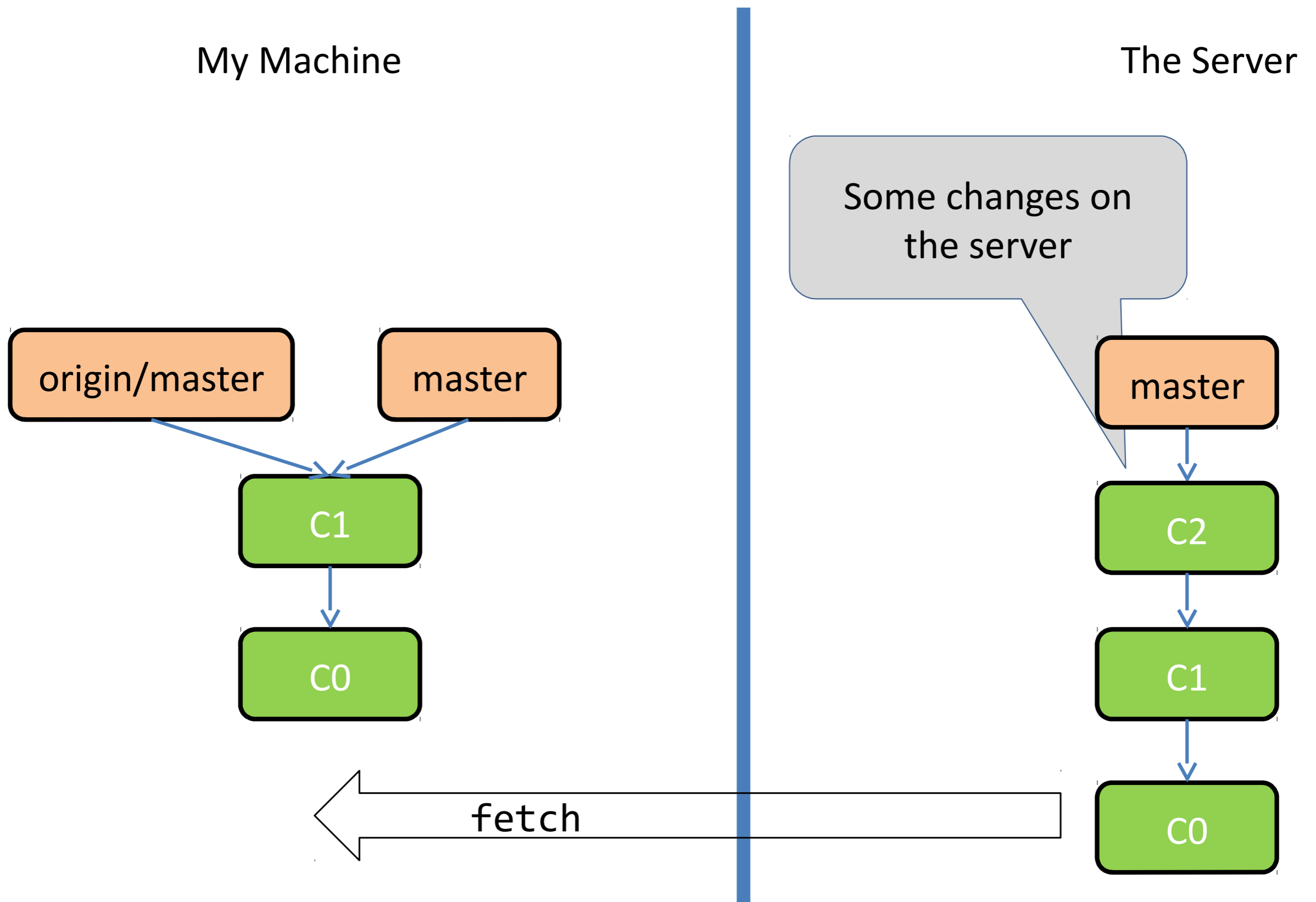
master

C1

C0

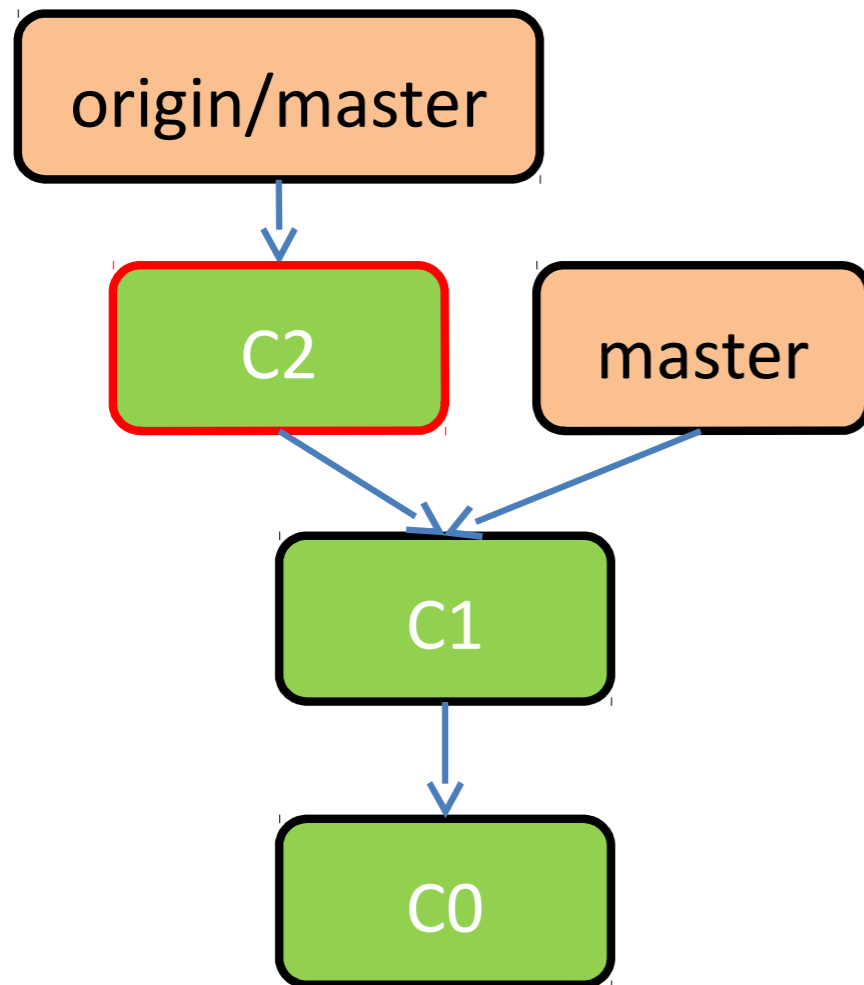


Remote Branches - fetch

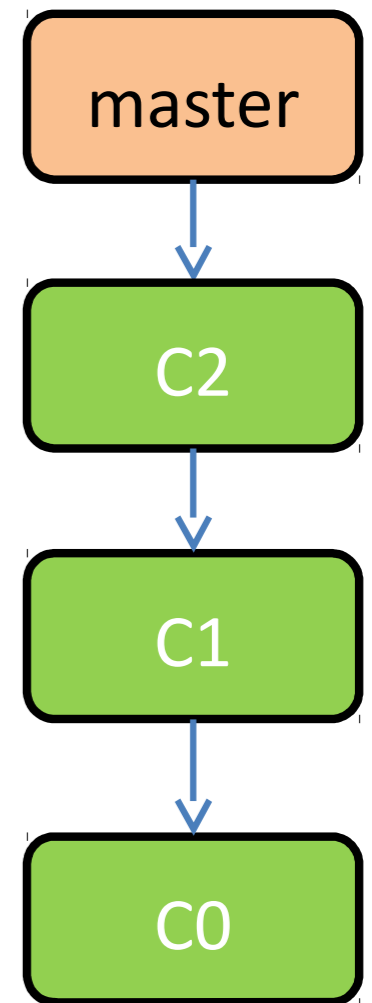


Remote Branches - fetch

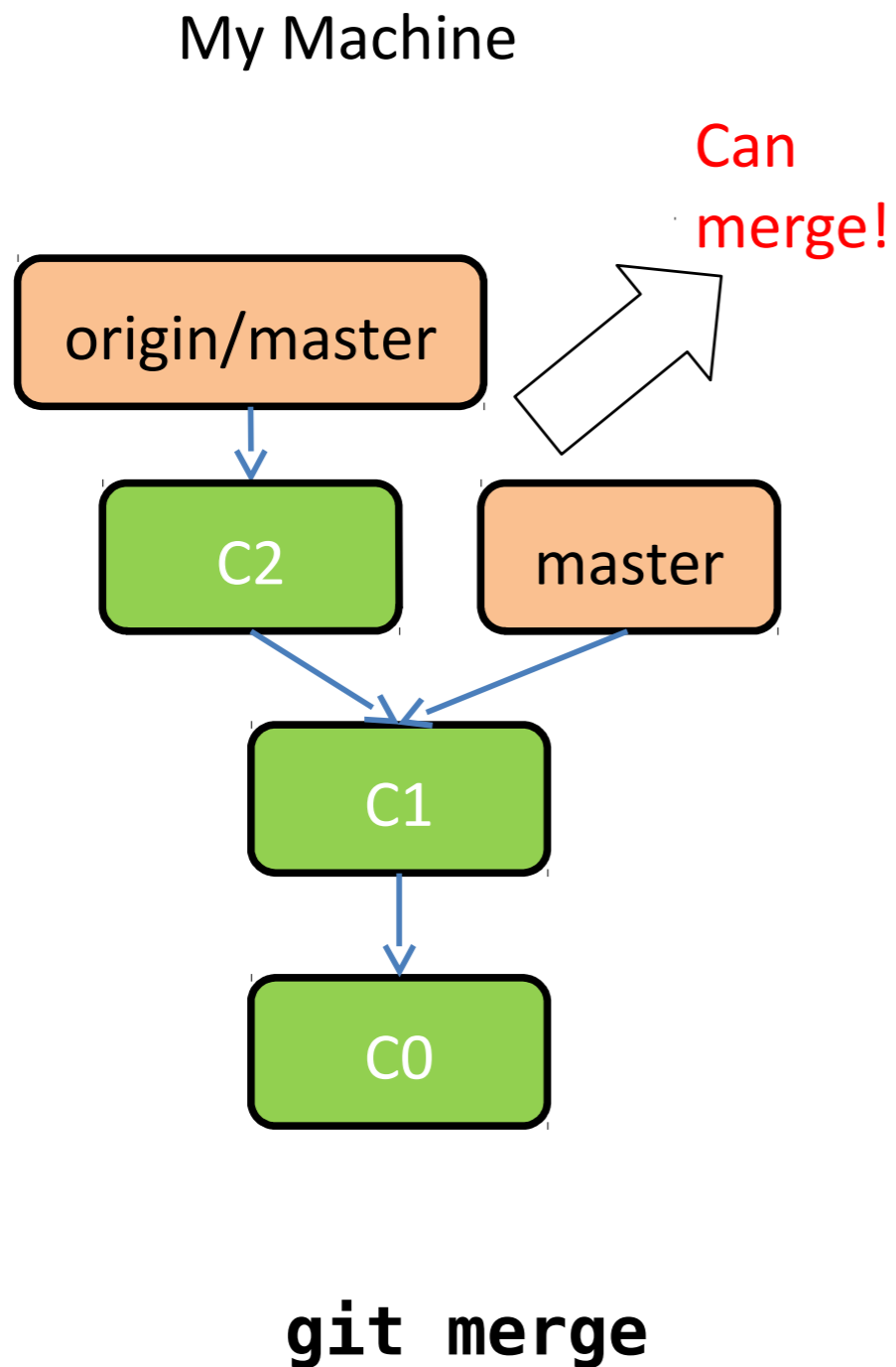
My Machine



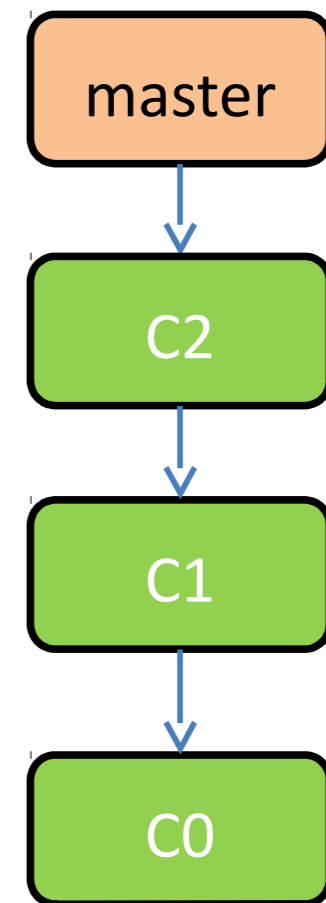
The Server



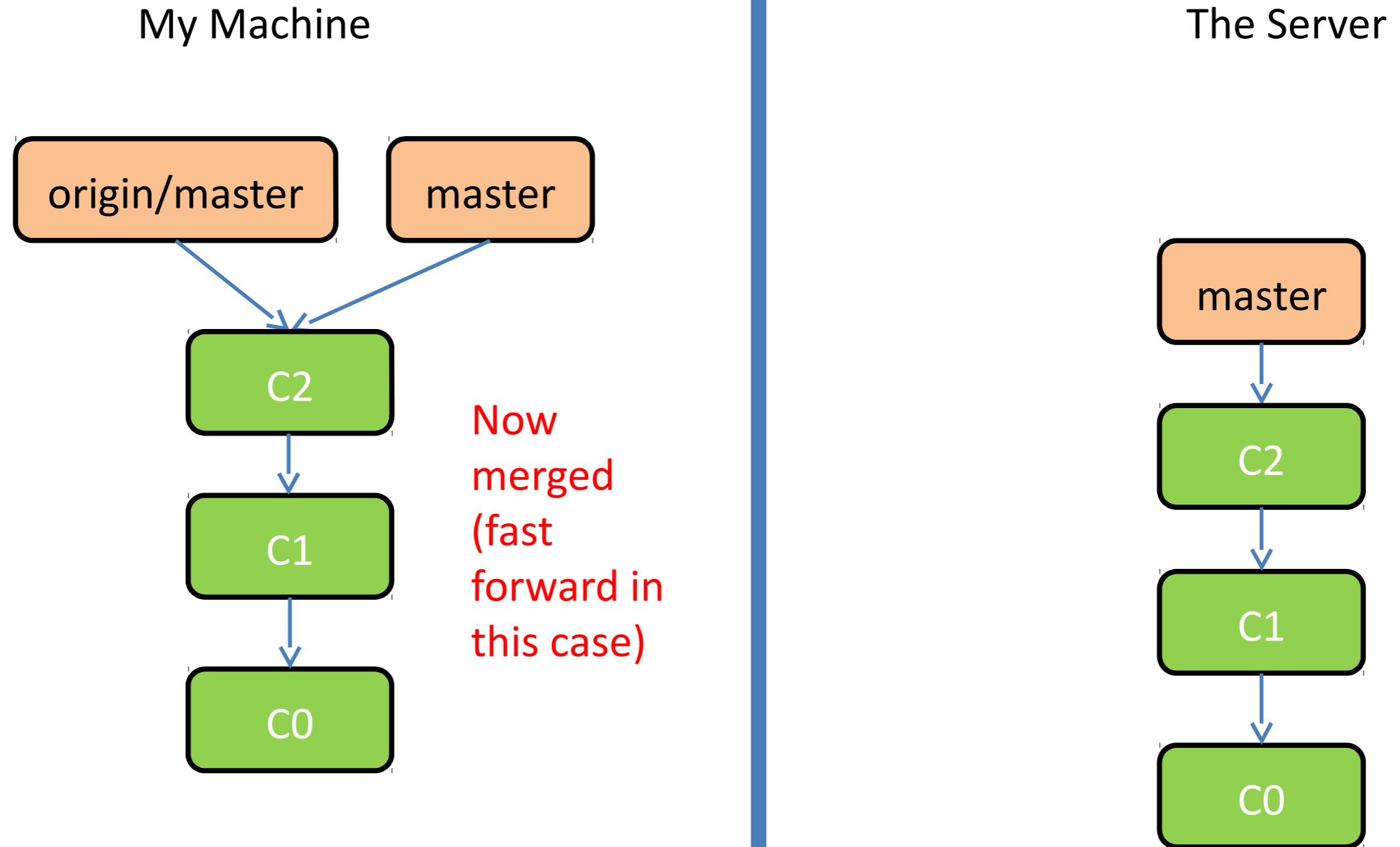
Remote Branches - fetch



The Server



Remote Branches - fetch



Remote Branches

- Reminder - Remote branches represent a branch on a remote repository
- The branch origin/master for example is a local pointer to the “master” on “origin”
- It reflects what the **local** repository **currently knows** about the state of “master” on “origin”
- You cannot change them, but you can “checkout” to get a “remote tracking branch”

Send information: push

- Will take local object which are required to make a remote branch complete and send them
- Will merge (fast-forward only) those local changes into the remote branch
- If fast-forward not possible:
 - the push will fail
 - need manual merge
 - `git fetch; git merge origin/master; git add .; git commit`

Conflict

Pushed on the server refused

```
$ git push origin master
To ssh://hall/~bcktestgit
! [rejected]      master -> master (fetch first)
error: failed to push some refs to 'ssh://hall/~bcktestgit'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

1) import the change from the server

```
$ git pull
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From ssh://hall/~bcktestgit
   a547735..7f32455 master -> origin/master
Auto-merging test.c
CONFLICT (content): Merge conflict in test.c
Automatic merge failed; fix conflicts and then commit the result.
```

Some change create conflict ! Need manual resolution

Conflict

Open the file(s) with conflict and resolve them

```
$ cat test.c
<<<<<< HEAD
line you wanted to push
=====
current version of the line on the server
>>>>>> 7f32455dbe6bea745bc94efd6b3d5f473446d581
$ vim test.c
```

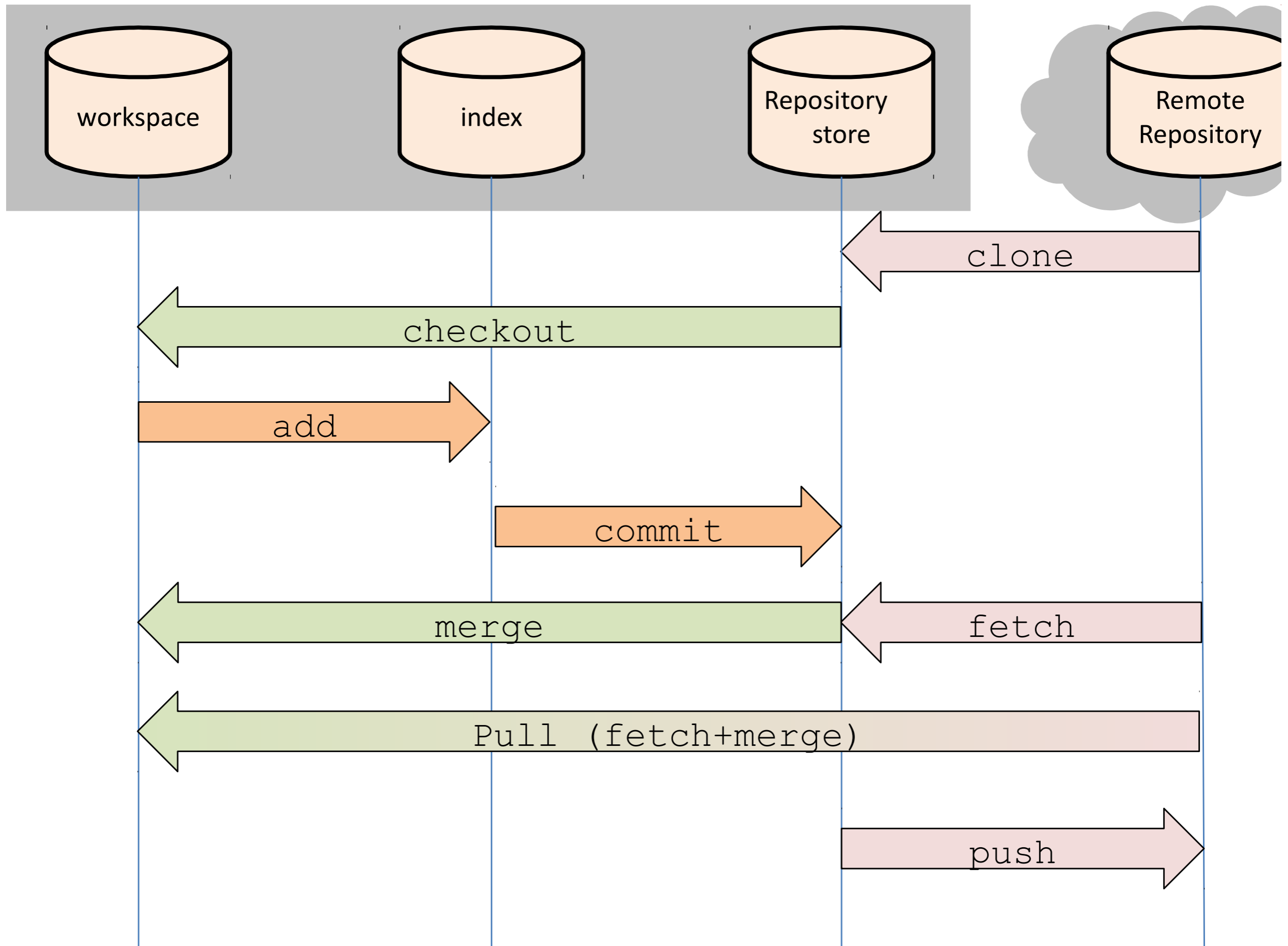
Commit your changes

```
$ git add .
$ git commit -m merge
[master 6b884f0] merge
```

Push on the server

```
$ git push origin master
Counting objects: 6, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 676 bytes | 0 bytes/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To ssh://hall/~bcktestgit
7f32455..6b884f0 master -> master
```

Summary of operations



Add your ssh keys!



Search or jump to...



[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)



Personal settings

[Profile](#)

[Account](#)

[Emails](#)

[Notifications](#)

[Billing](#)

SSH and GPG keys

[Security](#)

[Sessions](#)

[Blocked users](#)

[Repositories](#)

[Organizations](#)

[Saved replies](#)

[Applications](#)

SSH keys

[New SSH key](#)

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.



SSH

laptop

Fingerprint: 3a:e5:2b:68:2d:97:3a:b4:6d:74:47:25:01:84:09:44

Added on Jun 29, 2018

Last used within the last 4 months — Read/write

Delete



SSH

MBMGT

Fingerprint: 1a:0e:cb:fe:28:7a:fc:ca:8a:e3:06:9c:05:33:0f:30

Added on Sep 18, 2018

Never used — Read/write

Delete

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH Problems](#).

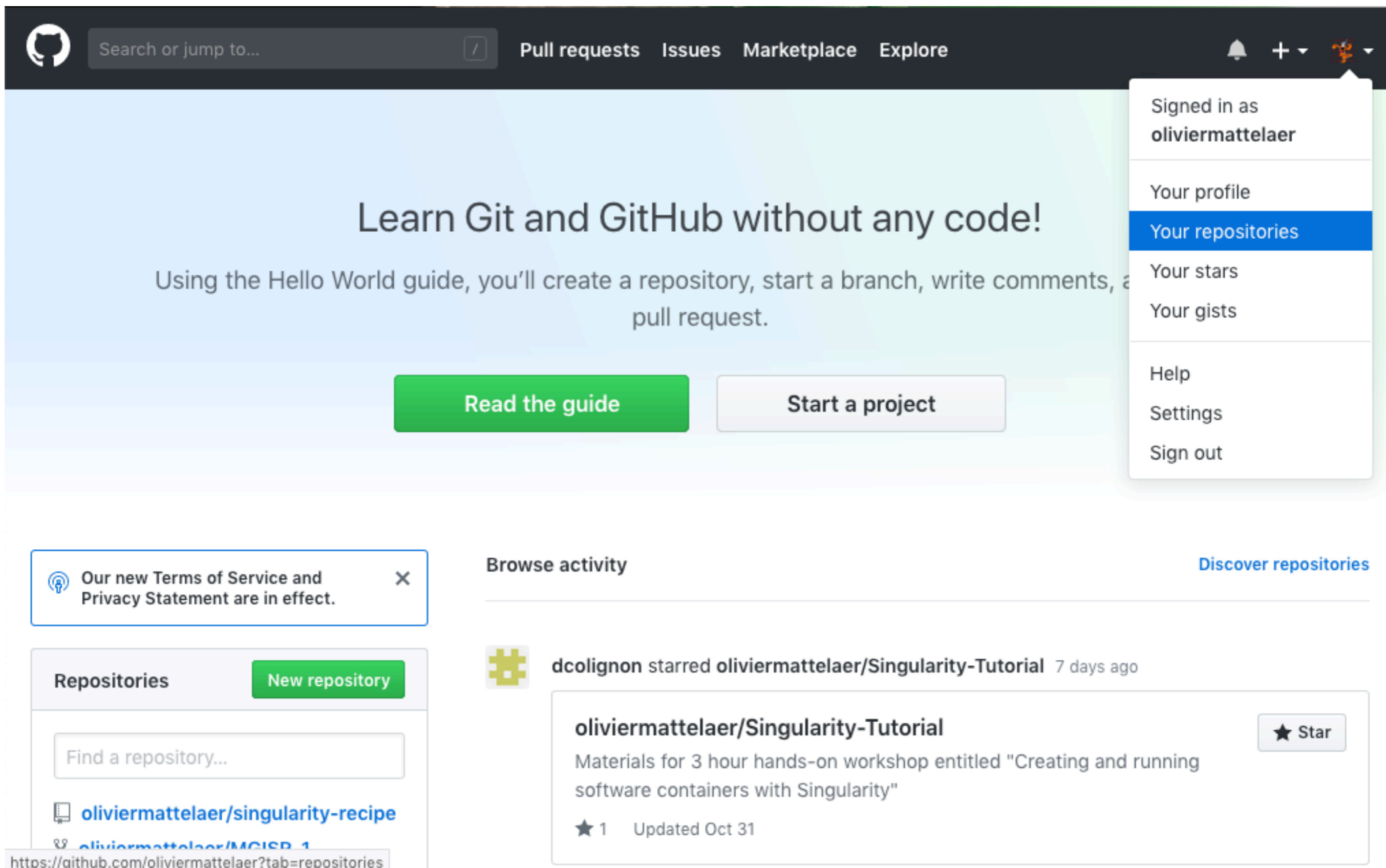
GPG keys

[New GPG key](#)

There are no GPG keys associated with your account.

Learn how to [generate a GPG key and add it to your account](#).

Add your project in git



The screenshot shows the GitHub homepage. At the top, there's a dark navigation bar with the GitHub logo, a search bar, and links for Pull requests, Issues, Marketplace, and Explore. A user menu is open on the right, showing the user is signed in as 'oliviermattelaer'. The main content area has a light blue background with the text 'Learn Git and GitHub without any code!' and a subtext 'Using the Hello World guide, you'll create a repository, start a branch, write comments, and pull request.' Below this are two buttons: 'Read the guide' (green) and 'Start a project' (light purple). On the left, there's a 'Repositories' sidebar with a 'New repository' button and a search bar. A notification banner at the bottom left states 'Our new Terms of Service and Privacy Statement are in effect.' The 'Browse activity' section shows a recent star by 'dcolignon' on the repository 'oliviermattelaer/Singularity-Tutorial'.

Signed in as **oliviermattelaer**

- Your profile
- Your repositories**
- Your stars
- Your gists
- Help
- Settings
- Sign out

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and pull request.

[Read the guide](#) [Start a project](#)

Our new Terms of Service and Privacy Statement are in effect. [×](#)

Repositories [New repository](#)

Find a repository...


[oliviermattelaer/singularity-recipe](#)

[oliviermattelaer/MCISP-1](#)

<https://github.com/oliviermattelaer?tab=repositories>

Browse activity

[Discover repositories](#)

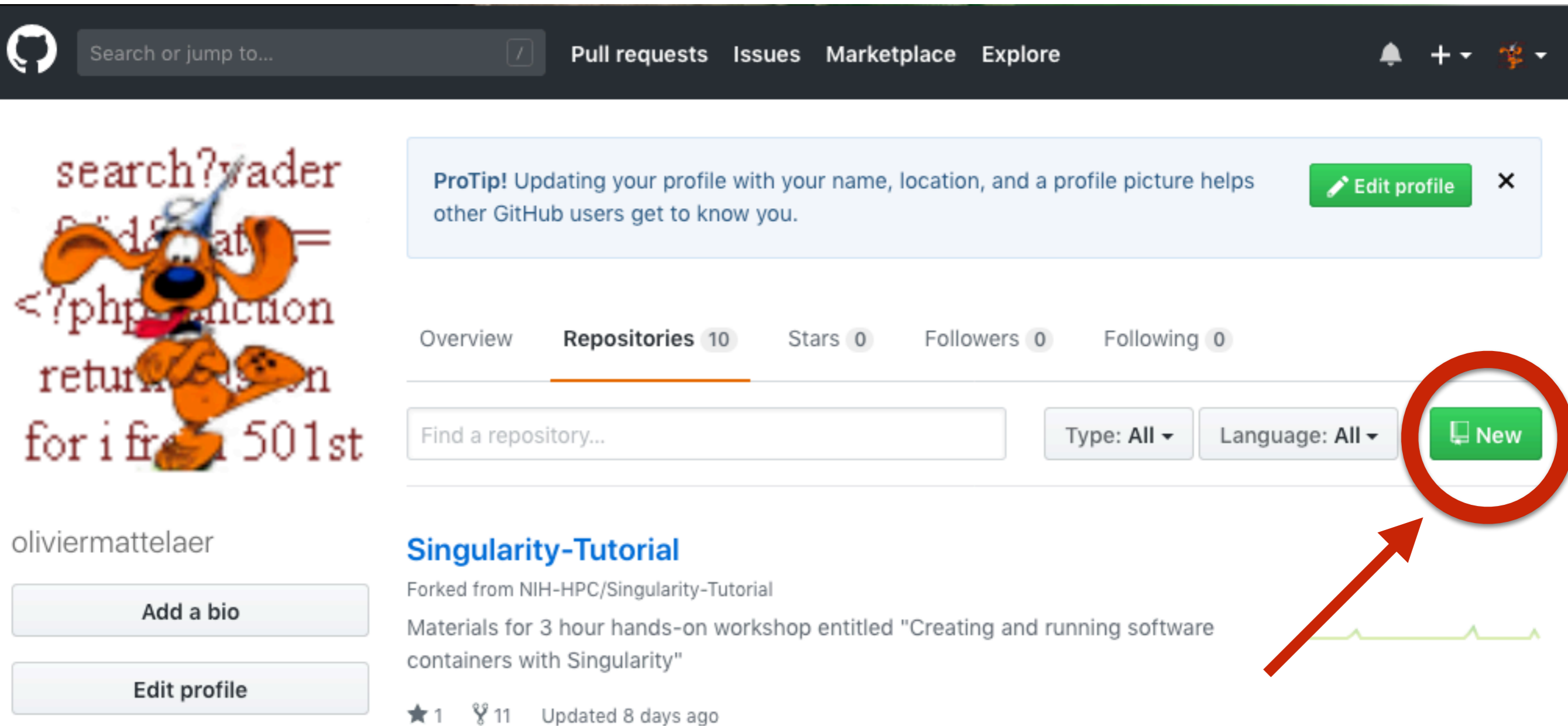
 **dcolignon** starred **oliviermattelaer/Singularity-Tutorial** 7 days ago

oliviermattelaer/Singularity-Tutorial [★ Star](#)

Materials for 3 hour hands-on workshop entitled "Creating and running software containers with Singularity"

★ 1 Updated Oct 31

Add it in a git repo



search?vader
P'd&at=
<?php function
return \$son
for i from 501st

oliviermattelaer

Add a bio

Edit profile

ProTip! Updating your profile with your name, location, and a profile picture helps other GitHub users get to know you. [Edit profile](#) X

Overview **Repositories 10** Stars 0 Followers 0 Following 0

Find a repository... Type: All Language: All **New**

Singularity-Tutorial
Forked from NIH-HPC/Singularity-Tutorial
Materials for 3 hour hands-on workshop entitled "Creating and running software containers with Singularity"
★ 1 🍴 11 Updated 8 days ago

Add it in a git repo

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

 oliviermattelaer ▾


Repository name

/ gittuto ✓

Great repository names are short and memorable. Need inspiration? How about **legendary-octo-happiness**.

Description (optional)

☒  **Public**
Anyone can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾



Create repository

Add it in a git repo

 oliviermattelaer / gittuto

 Watch 0

 Star 0

 Fork 0

 Code

 Issues 0

 Pull requests 0



 Projects 0

 Wiki

 Insights

 Settings

Quick setup — if you've done this kind of thing before

 Set up in Desktop or **HTTPS** **SSH** 

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).


...or create a new repository on the command line

```
echo "# gittuto" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/oliviermattelaer/gittuto.git
git push -u origin master
```

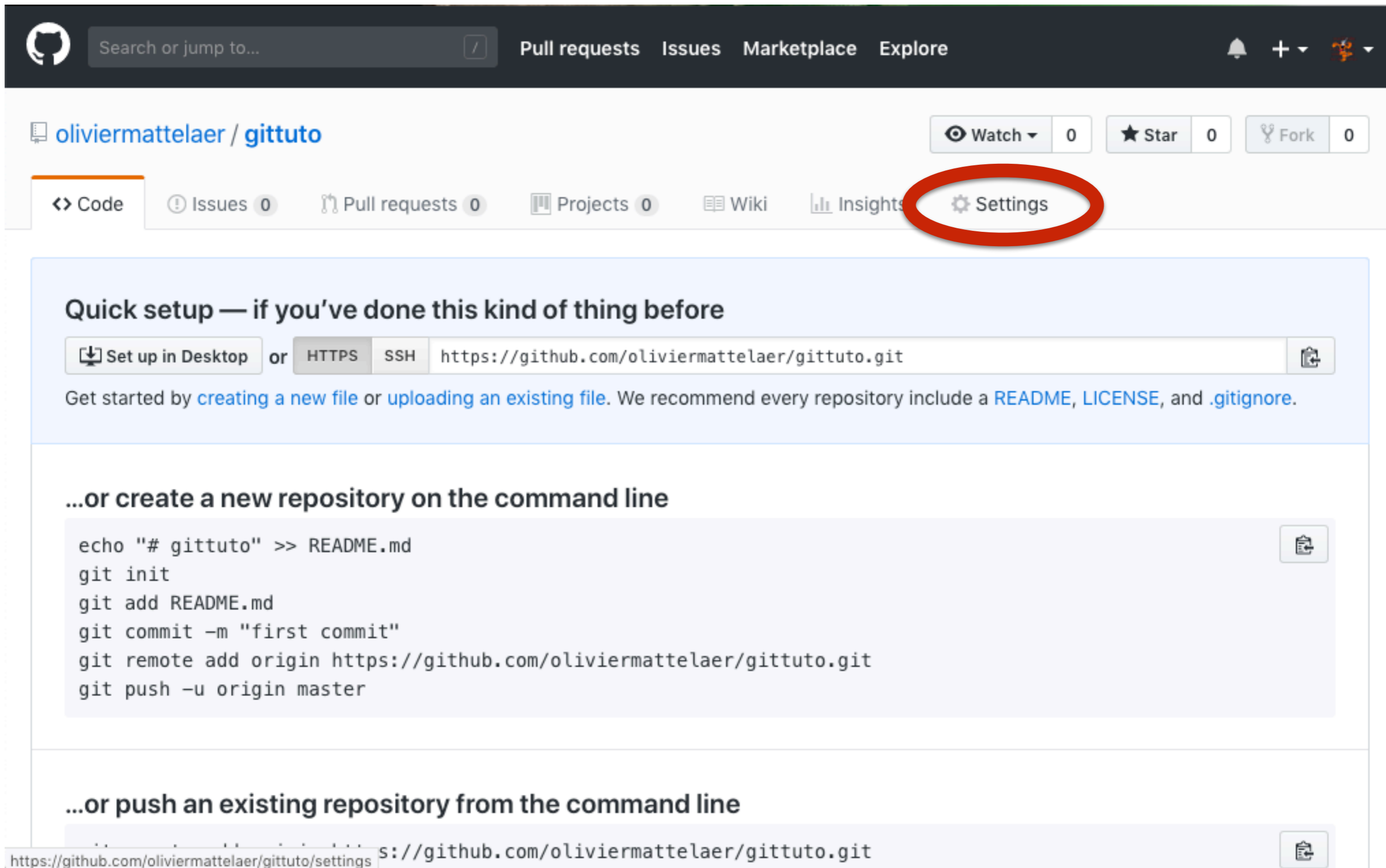


...or push an existing repository from the command line

```
git remote add origin https://github.com/oliviermattelaer/gittuto.git
git push -u origin master
```



Adding Collaborator to GitHub



The screenshot shows the GitHub interface for the repository 'oliviermattelaer / gittuto'. The top navigation bar includes the GitHub logo, a search bar, and links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. On the right, there are icons for notifications, a dropdown menu, and a user profile. Below the repository name, there are buttons for 'Watch', 'Star', and 'Fork', each with a count of 0. A secondary navigation bar contains links for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Insights', and 'Settings'. The 'Settings' link is circled in red. The main content area has a light blue background and contains instructions for setting up the repository. It includes a 'Quick setup' section with a 'Set up in Desktop' button and a text input field for the repository URL. Below this, there is a section for creating a new repository on the command line with a list of git commands. At the bottom, there is a section for pushing an existing repository from the command line with a git push command. The URL 'https://github.com/oliviermattelaer/gittuto/settings' is visible in the bottom left corner.

Search or jump to... / Pull requests Issues Marketplace Explore

oliviermattelaer / gittuto

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# gittuto" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/oliviermattelaer/gittuto.git
git push -u origin master
```

...or push an existing repository from the command line

```
git push https://github.com/oliviermattelaer/gittuto.git
```

<https://github.com/oliviermattelaer/gittuto/settings>

Adding Collaborator to GitHub

The screenshot shows the GitHub interface for a repository named 'gittuto' owned by 'oliviermattelaer'. The top navigation bar includes a search bar, links for Pull requests, Issues, Marketplace, and Explore, and user avatars. Below the repository name, there are buttons for Watch (0), Star (0), and Fork (0). A secondary navigation bar contains links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings (which is highlighted with an orange underline). On the left, a sidebar lists 'Options' (Collaborators, Webhooks, Integrations & services, Deploy keys) and 'Moderation' (Interaction limits). The main content area is titled 'Settings' and features a 'Repository name' section with a text input containing 'gittuto' and a 'Rename' button. Below this is a 'Features' section with three checked options: 'Wikis' (described as a simple way to let others contribute content), 'Restrict editing to collaborators only' (with a note that public wikis are still readable by everyone), and 'Issues' (described as integrating lightweight task tracking).

GitHub interface showing the repository settings for **oliviermattelaer / gittuto**.

The repository name is **gittuto**. A **Rename** button is available.

The **Settings** tab is selected, showing the following options:

- Options**
 - Collaborators
 - Webhooks
 - Integrations & services
 - Deploy keys
- Moderation**
 - Interaction limits

The **Features** section shows the following settings:

- ☒ **Wikis**
GitHub Wikis is a simple way to let others contribute content. Any GitHub user can create and edit pages to use for documentation, examples, support, or anything you wish.
- ☒ **Restrict editing to collaborators only**
Public wikis will still be readable by everyone.
- ☒ **Issues**
Issues integrate lightweight task tracking into your repository. Keep projects on track with issue labels and milestones,

Team project

Exercise #2

Todo List:

1. Create your account on GitHub
 1. Push your repository on github
2. Add a file “Authors” to your repository
3. Add a collaborator within GitHub
4. Import the repository of your collaborator locally
5. Add yourself to the Authors file of your collaborator
6. Sync “your” repository (and check that you have two authors inside now)
7. Create a conflict (if not faced already)

Miscellaneous



Adding backup

- Difficulty: Do not want to connect to the machine to force the update by hand
- Initialization on the cluster

```
$ ssh hall "mkdir -p bcktestgit && cd bcktestgit && git init --bare"  
Initialized empty Git repository in /home/pan/dfr/bcktestgit/
```

- Configure the path to the backup machine

```
$ git remote add hall ssh://hall/bcktestgit  
$ git remote -v  
hall ssh://hall/bcktestgit (fetch)  
hall ssh://hall/bcktestgit (push)
```

- push your branch on the machine

```
$ git push hall --all
```

Note: no working directory on the backup machine

Workflow/team work

- **web platform** are nice way to keep a team organized (gitlab, launchpad, ...)
 - provide common repository
 - provide visualisation of the branches/history
 - provide forum for discussion
 - merge request, bug report, questions
 - faq, targeted feature,...

branch organisation

master



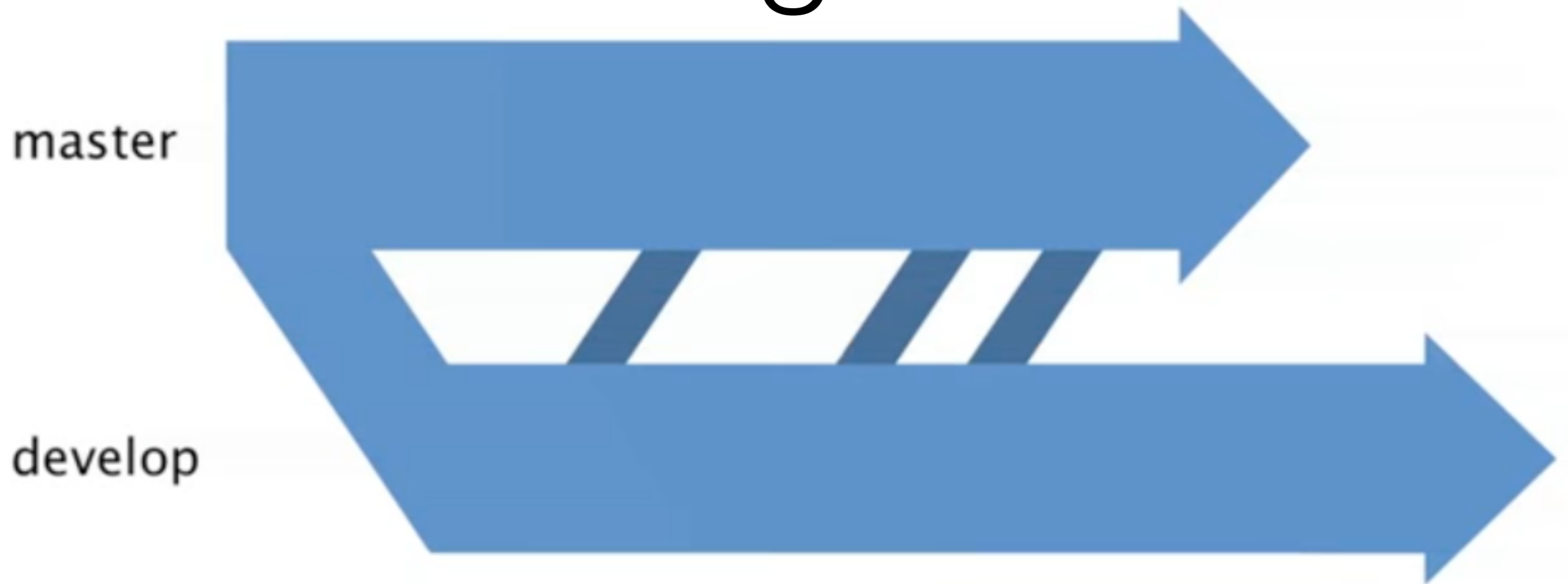
branch organisation



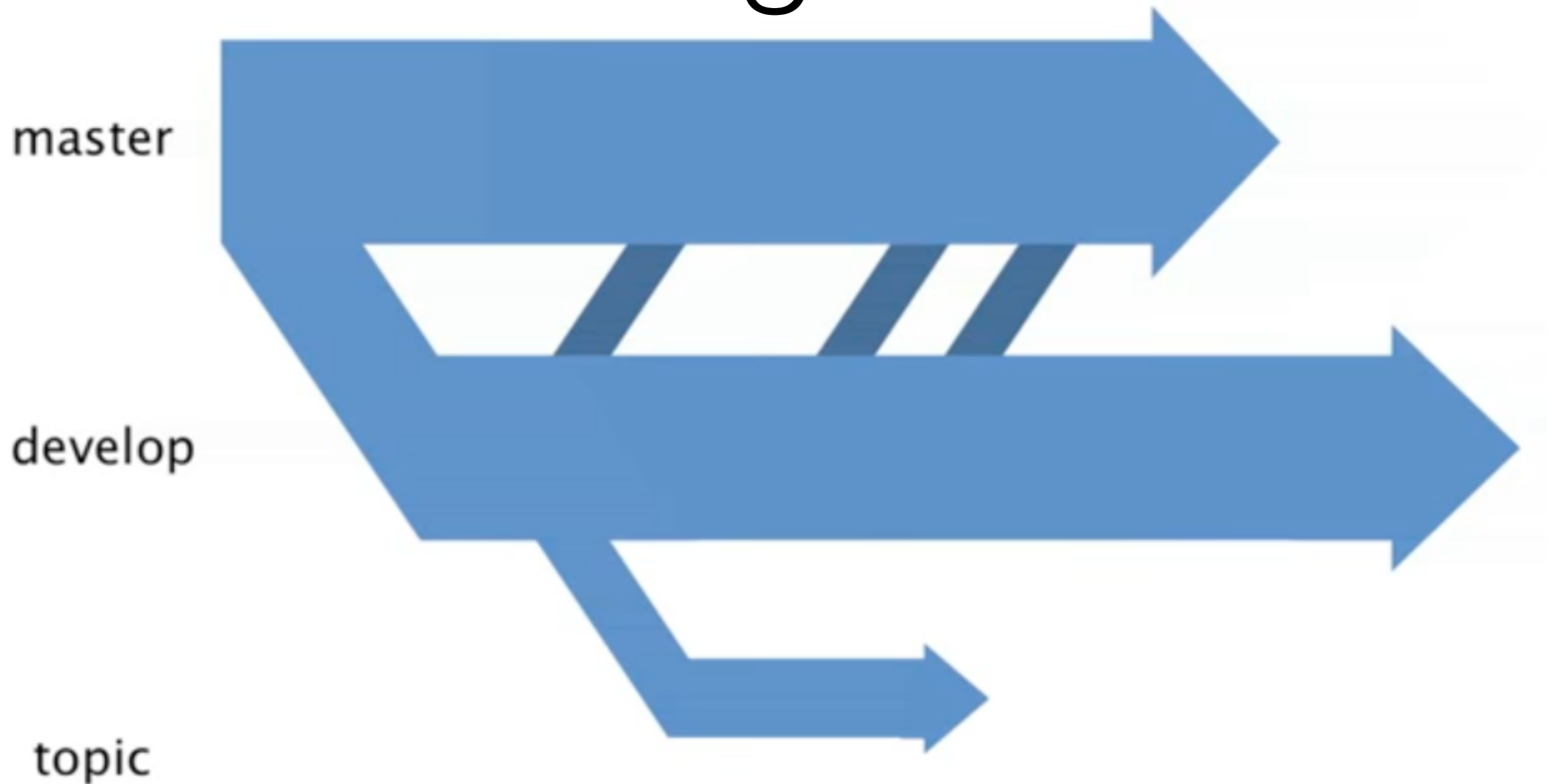
branch organisation



branch organisation



branch organisation



branch organisation



branch organisation



Conclusion

- Versioning is crucial both for small/large project
 - Avoid dropbox for paper / project
- make meaningful commit
 - logical block
 - meaningful message
- git more complicated but the standard

More information

- Why an index: <http://gitolite.com/uses-of-index.html>
- technical tutorial on git (details on storage structure): <https://www.youtube.com/watch?v=xbLVvrb2-fY>
- <https://git-scm.com/doc>